ECHO Users' Guide
for ECHO Participants

To accompany ECHO Version 5.5
April 2004

ECHO

Revision History

Version	Date	Description	Author
5.5	10/27/2003	metadata update example (604)	Ruowei Wu
5.5	10/28/2003	Metadata Ingest & Update: ECHO Actions (611)	Nancy Carney
5.5	4/12/2004	**Add echoToken text and code example (838):** Add new text on providing client identifier information in a user session. New Section 6.1.	Mark Nestler
5.5	2/2/2004	**Correct XML header for code examples (777):** Change all code snippets to use PUBLIC in the XML header instead of SYSTEM.	Bill Dickinson Jr.
5.5	12/18/2003	**Modify Section 5.1.10 of the Client Interface Guide (700):** Current text in section is confusing. Replace the text for the entire section.	Bill Dickinson Jr
5.5	12/18/2003	Content in Section 6.2 of the Client Interface Guide is unclear (701): Clarify content in Section 6.2 to remove confusion.	Bill Dickinson Jr.
5.5	10/31/2003	**Update Provider names (625):** Change the ProviderID references.	Nancy Carney
5.5	10/31/2003	Update UAS - user options (623): Change UAS option definitions.	Nancy Carney
5.5	8/15/2003	Errors in XML for ProviderOrderManagementService and ProviderProfileService (509, 510, 777): update headers in code examples.	Chris Dobbins
5.5	7/3/2003	backtrack add on in query and data type section (437): changes to Section 3.3.3.8.	Lei Fang
5.5	4/12/2004	Add text to Section 4.13.4 for Bug 802 and Bug 529 (840): Add text.	Eva Tu
5.5	2/2/2004	**Correct XML header for code examples (777):** Change all code snippets to use PUBLIC in the XML header instead of SYSTEM.	Bill Dickinson Jr
5.5	11/8/2003	**Metadata update (659):** Correct xpath in metadata update.	Ruowei Wu
5.5	11/6/2003	**DMS: temporal rules (645):** Change “Fixed Temporal” to “Temporal” and “Floating Temporal” to “Rolling Temporal.”	Nancy Carney
5.5	11/5/2003	**DMS rework (638):** Replace the 4th paragraph from end.	Nancy Carney
5.5	10/28/2003	**Renumbering Ingest Errors (613):** Modify the last paragraph in section 4.4.17.2.	Nancy Carney
5.5	10/28/2003	Metadata Ingest ECHO Granule/Collection DTD Spatial (612): Remove sentences referring to Oracle 9i.	Nancy Carney
5.5	10/27/2003	Changes to Metadata Ingest & Update Provider Duties (608): Replace third paragraph.	Nancy Carney

Version	Date	Description	Author
5.5	10/10/2003	**ECHO Browse Ingest Strategy (580):** Change <BrowseProduct> to <BrowseCrossReference>.	Nancy Carney
5.5	8/1/2003	**client identity (482):** Add client identity description.	Lei Fang
5.5	7/3/2003	**backtrack add on in query and data type section (437):** Modifications to Section 4.14 Spatial Coordinates & Projections.	Lei Fang
5.5	6/26/2003	5.5 user's guide for inspect functions (433): Minor editorial changes.	Ruowei Wu
5.5	6/2/04	Created new screen shots.	Bill Dickinson Jr.
5.5	4/29/04	Added text for Sections 4.1-4.3	Jim Amreihn
5.5	11/05/03	Modified draft into v5.5 official document	Bill Dickinson Jr.
5.5	6/30/04	Changed granule size in InpectDataset	Mark Nestler

Contents

Figures

Tables

Code Examples

Error Captions

Preface

The Earth Observing System (**E**OS) **C**learing **HO**use (**ECHO**) system went operational in November 2002 at Version 4.5 (Operational Release 1). This document describes the third operational release (Version 5.5). The ECHO system has evolved from a prototype to an operational system, and as such, will continue to evolve as user concerns are addressed. The evolution should largely be in the addition of new features, but may also involve changes to key parts of the system. These changes are largely derived from the data model review and have been approved for the system. The ECHO project will make every effort to summarize these changes so that client developers will have the opportunity to adapt quickly. Furthermore, the ECHO Development Team will try to get them worked in as fast as possible such that clients are impacted as little as possible.

This document is intended to provide the reader with approaches to using ECHO but not necessarily the specific details of calling every transaction. The online Application Program Interface (API) guide (http://api.echo.eos.nasa.gov/echo/message_detail.html) provides the details of how to create each message, but does not provide the big picture.

Conventions

All references to time are in Greenwich Mean Time (GMT).

1 INTRODUCTION TO ECHO

ECHO functions as a metadata clearinghouse and order broker for Earth Science Data and Information System (ESDIS) data and services applied to that data. ECHO hosts a cache of metadata representing the data holdings of a wide variety of providers. It adds value to existing systems by providing a single portal on the Internet where these metadata can be searched.

1.1 ECHO Concept and Design

1.1.1 General Architecture

ECHO has been constructed as a framework of components combined with a spatially enabled database that function together as a clearinghouse and order broker for Earth Science metadata. Internally, it specifies application process interfaces (APIs) and provides middleware components (including data and service search and access functions) in a layered architecture. This is further explained in the ECHO architecture document.

ECHO allows providers to cache copies of their metadata within it. Providers have complete control over what metadata is represented in ECHO on their behalf. They can insert new data, modify existing data and remove old data. Clients representing various entities can communicate with ECHO via its APIs in order to perform certain functions on ECHO's holdings.

All metadata is held in an Oracle database with spatial extensions. The metadata model is derived directly from that used by the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). Additional information that ECHO persists is also maintained in the database as persisted objects represented in a relational form.

Key features of the ECHO architecture are ease of provider participation, data model consistency, API specifications that enable direct user/scientist participation, the extensibility of user interface and an evolutionary development approach that can exploit industry innovations and change tack to incorporate user feedback.

Ease of Provider Participation: Designed to be low cost and minimally intrusive, ECHO offers a set of standard ways for providers to interface with the system and a metadata exchange approach that accommodates existing providers and technology (ECS Gateway, Data and Information Access Link [DIAL], V0 Gateway).

Data Model Consistency: In order to ensure data model consistency and ease of use in the ESDIS community, ECHO currently uses the Bulk Metadata Generation Tool (BMGT) format developed by ECS and is working with ESDIS and ECS to deliver a common format. To mitigate the risk of not being able to match all possible provider data models, ECHO has prototyped a Metadata Mapping Tool to translate non-standard formats upon ingest into ECHO.

Open System/Published APIs: ECHO uses an open system approach and ensures that user interfaces fully address user/scientist needs by specifying and publishing domain APIs that accommodate independent ECHO clients. These APIs are independent of the underlying transport protocols used. Currently, ECHO is capable of communicating using Simple Object Access Protocol (SOAP) and Java Remote Method Invocation (RMI). Plans are in place to add a web services view of ECHO services. Other transport protocols can be added as necessary. Interactions with ECHO may involve user interaction in real-time or may be machine-to-machine.

Extensibility of User Interface: ECHO extensibility is assured by its component architecture that allows new capabilities and functions to be plugged in, modeling relationships between services/APIs/UIs and continued prototyping. ECHO's current focus is on the middleware and enabling many different types of user interfaces via its APIs. Anticipated future development includes a componentized framework to support user interface customization.

Evolutionary Development: The ECHO system is developed in increments to allow for insight and feedback during the development cycle. Industry trends are followed and the use of commercial, off-the-shelf products is optimized.

1.1.1.1 ECHO as an API Framework

ECHO was designed to provide a message passing based infrastructure. The assumption is that the API functions by allowing an eXtensible Markup Language (XML) message to be passed to it, and then it responds with an XML message of its own. The messages passed in and received from the ECHO API interface conform to a set of Data Type Definitions (DTDs) that correspond to the particular messages involved. The DTD acts as a template, describing what a message should look like, but not every aspect. The DTD combined with some information about how to semantically interpret the message is enough information for an external entity to create messages to be passed to ECHO and to interpret its results.

ECHO provides a few functions that are applicable across all API accesses to the system and are therefore part of the framework. First, the ECHO framework provides an RMI-based interface. This interface defines three basic functions that an ECHO client can perform: login, logout, and perform a function (transaction) that is simply a way to pass an XML string in and receive an XML string as a result.

Second, ECHO provides a configurable translator system that converts an XML message into the appropriate Java method invocation. This translation allows the internals of ECHO to function in its native language speeding up processing. It also allows the addition, removal and modification of existing messages through a configuration process that does not involve coding, thus allowing for a more adaptable code base.

Third, ECHO groups transactions into services. Each service limits the types of users that are allowed to connect to it. For instance, those transactions associated with a provider updating its profile information (address, email, contacts, etc.) should only be executable by users that are logged in to the system in a provider role. There are currently guest users, registered users and providers. Each service has a fixed set of those user types that can use the service.

1.1.1.2 ECHO as a Spatially Enabled Metadata Search and Order System

Having defined a system that understands how to exchange XML messages, the next step is to have a system that understands spatially enabled Earth Science metadata. This is accomplished by introducing Oracle with its spatial extensions into the system and creating business logic within the system that understands how to interact with that metadata. In addition, a second interface to ECHO is added which allows metadata updates to go directly into the database bypassing the message passing API. A File Transfer Protocol (FTP) server is configured to receive these update files, which are also expressed in XML according to two DTDs, one for granules (or inventory) and one for collections (or datasets).

ECHO uses Oracle's spatial capabilities to search for data that has its spatial extent described within the system. A provider can associate boxes, circles and polygons with a granule or a collection. A client can then construct a search using these types of regions and ECHO responds with the data whose spatial region intersects the described region.

ECHO provides services for interacting with this catalog of metadata. Queries can be performed in a number of ways, result formats can be specified (what fields should be returned), and the resulting data sets can be incrementally accessed so that large return sets can be handled gracefully. ECHO also supports constructing, submitting and tracking orders for the data that the metadata represents. It supports both an embedding of a Uniform Resource Locator (URL) within the metadata for accessing the data (which the client simply accesses via Hypertext Transfer Protocol [HTTP]), or a more complicated order process in which quotes and order options are accommodated.

The query language must be embedded within a query transaction as a string and can be specified. This allows multiple query languages to be supported by the system. Currently only one is implemented which is a domain based query language built on XML and modeled after the V0 Object Description Language (ODL) queries.

1.1.1.3 ECHO as an Earth Science Metadata Search and Order System

The ECS model is used as a basis for ECHO, and therefore the ECS concept of granules and collections is incorporated. ECHO defines separate DTDs for updating each of these, and it is assumed that granules will indicate which collection they consider their primary collection. "Primary collection" means the collection that owns the granule – the collection that one would go to order that granule. Collections contain information that is common across all the granules they contain, as well as a template for describing product specific attributes (PSAs). PSAs allow a provider to define additional data fields in ECHO that encapsulate information that is not already part of the metadata model. Granules have their own metadata model, and additionally support the values that are associated with the PSAs defined by the owning collection.

1.1.1.4 Security

The ECHO system supports Secure Sockets Layer (SSL) based communication, which can be used by a client to pass passwords or other sensitive information securely. Internally, the systems are firewalled to prevent unintended access.

1.1.1.5 Supported Platforms

The ECHO system supports clients of any type capable of initiating an RMI or SOAP connection.

1.1.2 System Drivers

In order to effectively address the system vision outlined above, ECHO had also to respond to a set of system drivers. These drivers, derived from functional, organizational and operational concerns expressed by the user community, determined the architectural approach and the types of technical solutions used in building the system.

1.1.2.1 Ease of Participation

The primary goal of ECHO is to enable organizations to participate in making their resources and capabilities available to the Science community. In order to facilitate participation by these organizations, ECHO has:

Minimized the number of requirements that a provider must meet in order to participate. The only requirement for a provider to participate currently is to provide ECHO a copy of their metadata and work with ECHO to map it appropriately. Everything else is optional (implementation of search providers will change this).

Involved providers in the development cycle of the system and the definition of its requirements

Selected metadata insert and update mechanisms based on current standard industry practice (e.g., XML) that most databases can generate automatically

Provided mapping capabilities to convert from one XML representation into another

1.1.2.2 Cost to Field

While aggressive in the capabilities it is targeted to support, ECHO continually evaluates performance and functionality against costs in order to minimize the Cost to Field, e.g., licensing of COTS, amount of custom code required, hardware platform requirements and complexity of networking and installation.

1.1.2.3 Cost to Operate

Once fielded, ECHO seeks to minimize the costs to operate the system. Every effort is made to minimize requirements for operations staff, both in skill and in quantity.

1.1.2.4 Cost to Maintain

ECHO's architecture, development processes and use of commercial products have all been selected to minimize the costs involved in maintaining the system, e.g., programming staff required for evolutionary improvements, maintenance costs of COTS products, potential functional enhancements.

1.1.2.5 Extensibility

ECHO is being built with long-term extensibility foremost in mind. In order to enable emerging techniques and strategies for Earth science research, ECHO has:

Adopted 'design for change' as a goal at the beginning of ECHO development

Built in the capability to limit the impact of changes to the API to one configuration file, the service interface and the business logic that actually implements the function

Developed test tools to regression test large portions of the functionality automatically overnight, so when changes are made, non-desired impacts can be discovered quickly

Adopted a layered ECHO architecture to allow changes to one component of the system without affecting other components

1.1.3 ECHO Capability/Functionality

ECHO functions as a metadata clearinghouse and order broker for ESDIS data and services applied to that data. ECHO hosts a cache of metadata representing the data holdings of a wide variety of providers. It adds value to existing systems by providing a single portal on the Internet where these metadata can be searched.

ECHO provides an infrastructure that allows various communities to share tools, services and metadata. As a metadata clearinghouse, it supports old and new data access paradigms such as navigation and discovery. As an order broker, it forwards orders for data discovered through the metadata search process to the providers for satisfaction. As a service broker, ECHO decentralizes end user functionality and supports interoperability of distributed functions.

ECHO supports providers at three different levels: Data Provider, Service Provider and Search Provider. These different levels of participation are not exclusive.

Data Providers – organizations that supply inventory-level metadata representing their data holdings to ECHO. ECHO offers well-defined interfaces and reusable tools to participating providers in order to accomplish these goals with minimum investment of resources and effort, on the provider's part.

Service Providers – organizations that participate by advertising their Earth Science related services to the user community via ECHO. ECHO will maintain service descriptions in a service catalog (either special services, or services that are available as an option on a selected set of granules/collections) and support the user in ordering those services.

Client Providers – organizations that participate by sharing a "catalog" of the types of data, or collections, that they manage. These providers will participate in Distributed Searches by receiving queries, resolving the queries on their own processing resources and returning the results of those queries to users through ECHO.

ECHO addresses Science User needs through a set of well-defined and open interfaces upon which the user community can build their own client applications. In this way, the ECHO supports extendable, flexible user interfaces, allowing industry and the community to drive the progress of available Earth Science applications.

Groups outside the ESDIS community can also subscribe to metadata holdings in order to build their own systems. The ECHO approach allows users to build their own user interfaces (clients) to ECHO so they are not limited by the data search and order system provided by NASA. For data providers, ECHO off-loads the burden of providing the system resources required for searching and gives them the flexibility to support community-specific services and functionality. ECHO's interoperability features allow all participants to benefit from the distributed development of functions, again reducing dependence on NASA resources.

1.2 How to Talk to ECHO

One of the system drivers in designing ECHO is that there should be programmatic ways to access the system. In other words, the system should not rely only on a Graphic User Interface (GUI) for user access. All functions of the system can be accessed programmatically. Specifically, a user of the system need not be a person, but could be another computer. This focus has led to the API interface being developed without a coincident user interface. The intent is to allow many varied user interfaces, each of which addresses its own community of users. It should also be pointed out that while the API has been defined to make some things easy for GUIs, it is not responsible for the end user interface's look and feel and therefore doesn't attempt to make everything simple to use.

1.2.1 Message-Based APIs

The majority of all interactions with the system take place through a message based API. The basis of these messages is XML. XML provides a structure and syntax, but it does not specify the semantics of a message. ECHO accepts a message in XML, performs the requested transaction, and then responds with any results and status in XML as well. The messages are passed into ECHO currently through Java Remote Method Invocation (RMI).

With the exception of session management (logging in and out), all interactions are simply sending an XML string and receiving one in return. Because of the textual nature of this interaction, the protocol used to exchange messages is not as important.

As noted above, the exception to this is session management. It is important to have a method of establishing a session as a known user and then disconnecting from it. In the current ECHO implementation, there are additional RMI method calls for login and logout. If other protocols are used, then some secure method for establishing a session must be created.

The API transactions are divided into groups known as services. Each service has some number of transactions that are grouped according to their actions and only allows users of certain types to connect to it.

1.2.1.1 The Session Manager

All communication with ECHO takes place through the Session Manager. A client uses a SOAP client library to obtain a reference to ECHO's Session Manager, and then sends all communication to ECHO through that reference. The Session Manager has a very simple interface:

Table 1-1: Session Manager interface

Method Summary		
void	identify(String client)	Used to identify the client accessing ECHO. All clients should call this after establishing a session.
boolean	login(String username, String password)	Used to establish a registered user session.
void	logout()	Used to end a registered user session.
String	perform(String msg)	Access to ECHO's XML message based services.
boolean	setProviderContext(String providerId)	Used by a registered user to specify which provider they are acting as (if they can represent more than one).

A client, once it has obtained a reference to the Session Manager, should identify itself through the identify method. The string submitted should be a brief identifier for the client and its version number (e.g., EDG-E v1.3). Once this is done, perform can be used to submit XML messages for service requests that match the ECHO API, and the resulting XML message is returned from it. If login has not been called, then the client is acting as a guest user. If login has been called, then the client is acting as that user. It is expected that the client will simply provide a userid and password prompt for its user, and then pass that information to ECHO. Logout can be used to end a registered user's session. SetProviderContext is used when performing provider maintenance functions to identify which provider a registered user is working on behalf of if that registered user has more than one provider role (See roles description in section 2.4). If there is only one provider role for the registered user, then it is not necessary to call this function, as ECHO knows which provider to reference. If there are no provider roles for a registered user, then that user cannot execute the provider management functions.

1.2.1.2 XML Basics

In order to understand ECHO as a developer using it, it is important to have a fundamental understanding of ECHO. The ECHO Development team has used the WROX series of books and found them satisfactory as learning and reference material. This section strives to provide a modicum of information needed to get started with ECHO and is not intended to be an XML tutorial.

Note: If one decides to use the Java Client Library/Façade, then this level of complexity is hidden from the client.

ECHO uses DTDs as the templates for the messages that it handles. These templates can be used to help construct a message the first time, and to validate that a message is constructed correctly. The basic structure of an ECHO API message is:

```
<ServiceName><TransactionName><TransactionSpecificParameter>…</TransactionSpecificPara
meter></TransactionName></ServiceName>
```

Note that the XML structure is similar to Hypertext Markup Language (HTML). Each part begins with an open tag inside of angle brackets, and ends with the same tag with a leading slash. The first level tells what individual service of ECHO is being addressed. The second level identifies the particular transaction that is being requested. If there are no parameters, then that is all that is needed, otherwise the parameters are embedded inside at this level according to the appropriate DTD.

1.2.1.2.1 XML Escape Characters

Providers must supply a valid XML file for their metadata. A metadata input XML file should be validated against its corresponding DTD before being sent to ECHO. Certain characters such as "&" etc. in the XML file must be addressed. The input string containing those characters can be either expressed using CDATA expression or using escape characters.

The following examples shown how the string should be given in the XML file:

for long name:

```
<name part 1 & name part 2>
```

using CDATA expressions:

```
<LongName>![CDATA[<name part 1 & name part 2>]]</LongName>
```

using escape characters:

```
<LongName>&lt;name part 1 &amp; name part 2&gt;</LongName>
```

The ECHO ingest process handles the escape characters in Table 1-2 if they appear in the input XML file.

Table 1-2: XML escape characters for ingest.

Character	Escape Character
&	&amp;
<	&lt;
>	&gt;
'	&apos;
"	&quot;

1.2.2 Metadata Ingest and Update

FTP is used to transfer metadata ingest files. This streamlines the providers' process of putting data into ECHO. Three XML DTDs – one for collections, one for granules, and one for browse – define a template for these updates. Each DTD contains three optional sections: add, update and delete. In the current implementation, there is no difference between add and update. If a granule that already exists in ECHO appears in the "add" section, the granule is simply replaced with the new incoming information. If a granule that is not currently in ECHO appears in the "update" section, that granule is added to the system.

Some providers have data pools. They update the data pools frequently; the information involved typically includes Online Access URL and Quality Assurance (QA) Flags. The data provider may update those elements for any existing granule. A metadata update DTD is defined to help with this.

Metadata ingest and metadata update files are slightly different. Metadata ingest files – both for insertion or replacement – require the complete metadata information associated with a collection or granule. Metadata update files require only granuleUniversal Reference (UR) and data pool information such as Online Access URL and/or QA flags. The granuleUR referred to in the metadata update file must be registered with ECHO, or the request will be rejected.

Note: The metadata update process is only available for granules at this time.

1.2.2.1 Provider Duties

The Provider is responsible for creating an XML file that represents the metadata of their system. The provider might create a database query that extracts all new or changed data from their database since a certain date/time. By keeping track of the date/time of the previous query as part of the new query, incremental changes can be extracted.

Next, the Provider is expected to convert the resulting metadata into an XML format. Many databases have this ability built in, but scripts can be written to do the text formatting necessary to perform the conversion. ***If possible, putting the metadata directly into the format dictated by ECHO's DTDs is desirable.*** Once an XML file that conforms to ECHO's DTDs is available, the provider simply FTPs the file to a known directory on the ECHO operational machines. This is currently done through a named FTP account.

If a provider's XML version of metadata does not match the format described by the ECHO DTD, the provider must write a utility to translate his or her XML to conform to the ECHO DTD. XSLT and Java support classes are the recommended mechanisms for developing this translation. The provider's translation script must be incorporated into the ECHO runtime environment. Providers should contact the ECHO Operations Group for assistance in this matter.

Similarly to metadata ingest, the provider is responsible for creating XML files that represent metadata update. However, the XML files should conform to the update metadata DTD of ECHO.

1.2.2.2 ECHO Actions

Every five minutes, ECHO checks for metadata update files. Once a file is detected, it is validated (see below), and then copied into the FTP input file processing directory for processing. If the input XML file is based on a non-ECHO DTD, then ECHO uses the provider-supplied conversion utility (discussed in *Section 1.2.2.1 – Provider Duties*) to convert the file to conform to the ECHO DTD.

The ingest process validates the input file in three ways. It validates:

1. Whether the file is an XML file. If the input file is not an XML file, then this file will be removed from the input directory and the error will be recorded.
2. Whether the file is an appropriate XML file for this directory (for instance, a collection XML file should be deposited in the FTP incoming directory for collections). If the file has been placed in the wrong directory, this file will be removed from the input directory and an error will be recorded.
3. Whether the input file conforms to its corresponding DTD. If the input file does not conform to its corresponding DTD for any reason, the file will be removed from the input directory and an error will be recorded. Files containing errors will not be processed.

The ingest process also examines granules for specific characteristics and does not ingest a granule if:

1. A granule is not associated with a collection already in ECHO;
2. Spatial metadata associated with the granule is invalid.

When ingest completes, two reports are sent to the provider's registered email address:

1. A report with the error information on the problematic files . This report is written as an XML file with reference to the DTD of ErrorInputFileSummaryReport.dtd
2. A report showing how many granules and collections were ingested, how many were rejected, and the reasons for the rejection. The report is written as an XML file with reference to the DTD of ingestSummaryReport.dtd.

2 ECHO ENTITIES

Interactions with ECHO are done through message passing and affect the state of entities within the system. The major ones are described in this section.

2.1 Users

The most basic entity in the ECHO system is a User. Users can be distinguished from each other based on their username, which is unique to any particular user of the system. There are two types of users: registered users and guests. Guests have the ability to do many of the things registered users can do, but they cannot count on persistent access to information across sessions. Registered users can save information to be used in their next session after they log out of the current one. Registered users can further be segregated into Providers and Customers. Provider users are associated with a data center. Customers are associated with an end client user.

2.1.1 Addresses

Customers (guest or registered user) are not required to submit address information to the ECHO system unless they order a piece of data that requires a shipping address, a billing address or a contact address. A registered user has the capability of adding any number of addresses into their profile but is not required to do so. An ECHO Address entity consists of an address ID, a US format flag, five Street Address lines, and city, state, zip code, and country fields. The address ID represents a unique name for the address entity like "Home" or "Work". The ECHO Address entity does try to support multiple international mailing formats by using the US format flag. Since, international mailing formats can differ greatly from each other, the ECHO Address entity has five Street Address lines that allow for a free format that users and providers can use to fill in any international address in any way that is deemed best. In such a case, the US format flag should be set to “FALSE” and only the first street address and country field are required. However, if the mailing address does follow the normal U.S. mailing standards, then the US format flag should be set to “TRUE” and the city/state/zip-code/country fields will become required fields. In all cases, the country field is required and must comply with the ISO 3-letter country code. For example, “USA” stands for the United States of America while “CAN” stand for Canada.

2.1.2 Emails

Registered users of the system are required to submit an email address to ECHO during registration. This is different from a guest user, which is not required to have an email address. The rules for an email address in the ECHO system are consistent with global standards for email addresses: username@somedomain.

2.1.3 Phone Numbers

Phone Number entities are similar to Address entities in that a user may store multiple phone numbers in the ECHO system. Each piece of phone information is given a unique name chosen by the user. The PhoneInformation entity consists of a PhoneName, a country code, area code, exchange code, phone, and extension (optional). An example of a PhoneName is "Home" or "Work".

2.1.4 Preferences

User's can specify default "preferences" which are used by ECHO during the ordering process and are implemented using the ECHO’s properties framework. This is also used in ECHO’s provider policies and order options. An example of a user preference would describe something like ‘only ship packages via FedEx 2nd day’. Preferences, as well as other types of ECHO properties are a complex topic, and separately covered in *Section 2.3 – Properties and Options*. Currently, no user preferences have been defined to date.

2.2 Providers

2.2.1 Provider contacts

In order for a Provider to register with the ECHO system, they need to specify one or more “provider contacts.” A Provider Contact represents a person with the provider’s organization. A Provider Contact has a “Contact Role”, which is a unique identifier and represents the title or position of the person. A good example of a “Contact Role” would be “order manager”. Specifically in ECHO, any provider contact with the Contact Role of “order manager” is responsible for receiving copies of the email sent to the users for order status updates and thus able to answer questions from the users about their orders. Provider contact entity contains: FirstName, LastName, AddressInformation (optional), PhoneInformation (optional), and an EmailAddress. Providing an EmailAddress is important in order to receive copies of any email notifications to users or the providers themselves.

2.3 Properties and Options

The ECHO application frameworks include a mechanism that allows properties to be dynamically defined and attached to entities within the system. A property is a named value, similar to an attribute. The "properties" mechanism has been created to enable easy, code-free extensibility of ECHO entities. Order and User Profile objects currently rely upon this properties mechanism to enable portions of their state to be dynamically defined and set.

The properties mechanism is useful for managing properties that are defined and used by systems that are outside of the context of ECHO. A data provider that ECHO interacts with is a good example of such an external system. The data provider may require that specific information accompany any order or request for data. Furthermore, portions of that information may be uniquely specific to requests for data sent to that particular provider. This is a situation where applying the property mechanism makes sense. Here, the provider-specific information will be defined and stored abstractly as a collection of properties that must be set by the user during the order entry process.

The ECHO API uses the parameters Option and OptionSelection to communicate property definitions and property values respectively. An Option parameter defines a property value that may be set through the ECHO API. An OptionSelection parameter is used to set the value that is defined by the option or to communicate the value that is currently set for the option. The diagram in Figure 1 shows the structure and relationships of the option parameters, as defined in the XML API.

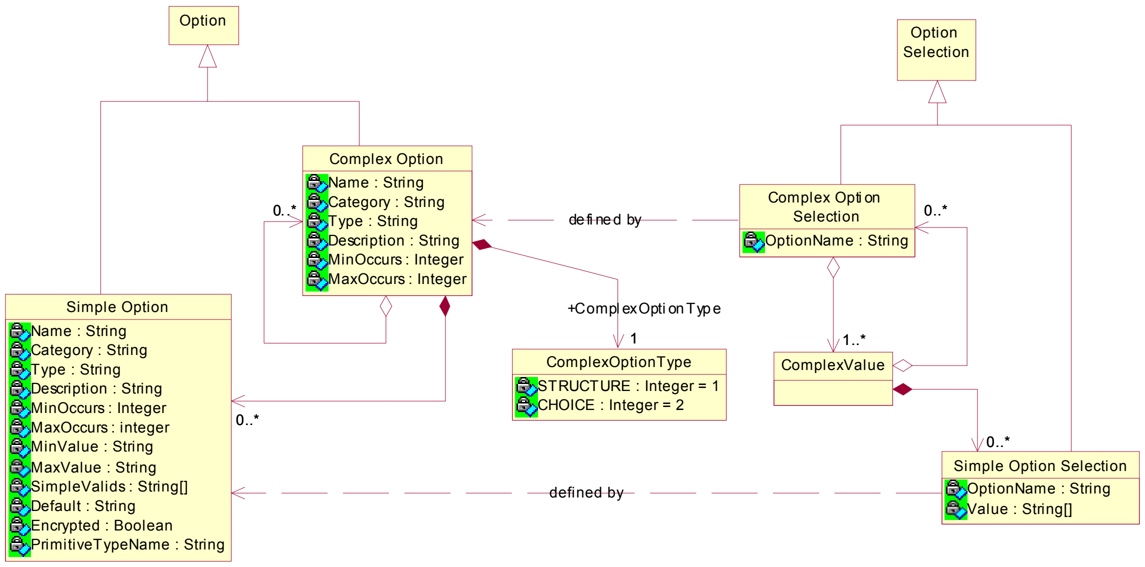


Figure 1-1.Structure and relationships of the Option Parameters.

There are two types of options: simple and complex. A simple option describes an atomic piece of information that of a primitive type. The "Type" field indicates the primitive value type that is expected. A simple option may also specify that multiple instances of that type may be input--similar to an array. The "MinOccurs" and "MaxOccurs" field values define the multiplicity of a corresponding value, while the MinValue and MaxValue fields define the range of allowable values. A simple option may declare the complete set of distinct valid values that may be set for a simple option selection of that type. These are defined in the "Valids" field. Finally, a default value may be specified in the "Default" field.

The second type of option is ComplexOption. A complex option is an option that defines a grouping of input parameters, sub-options, and/or choices. Like simple options, complex options may declare the allowable multiplicity of values. These are defined in the "MinOccurs" and "MaxOccurs" fields. The complex option "Type" field indicates the validation behavior that will be applied to the corresponding complex value instance. There are two types of complex value: structure and choice. A "choice" type indicates that for each complex option selection value (remember that there may be many values), a sub-value for only one of the sub-options contained by the complex option may be set. A "structure" type, on the other hand, allows a complex value to be set that contains set sub-values for each sub-option defined by the complex option.

An OptionSelection is the actual that corresponds to an Option. Mirroring the option hierarchy, there are two types of option selection: SimpleOptionSelection, and ComplexOptionSelection. A simple option selection has a name that matches the name of its simple option, and contains one or more primitive values. A complex option selection also has a name that matches the name of its [complex] option, and contains one ore more complex values. A complex value is a container of option selections. It will contain zero or more simple options and complex options, as defined by the complex option and its associated "Type."

Due to the reflexive nature of options and option selections, reading and generating appropriate OptionSelections can be confusing. An example definition with some explanations and mapping between the definition and the selection has been provided.

2.4 Roles

In the past, there was a concrete concept of two kinds of registered users that could log into the system – the science user and the provider user. Now, only one kind of user can log into the system – the science user. All authorization in the system has moved from the service layers to the transactions themselves. Provider-oriented transactions are no longer used by provider users but instead facilitated through the concept of provider roles that are associated with a science user.

ECHO now has a concept of "user roles". User roles are a way to grant a user access to the system. These roles facilitate greater flexibility with transaction-level authorization and allow certain users the ability to execute both user and provider transactions without having two accounts in the system. Currently, there are two kinds of roles:

Provider Role: A user can have one or more provider roles, with each role associated with one provider in the system. A user with one or more provider roles can access and update information about the providers with which they are associated. For instance, if a user has a provider role for Oak Ridge National Laboratory (ORNL), then that user can use the UpdateContact transaction in ProviderAccountService to update the contact information for ORNL.

Admin Role: A user can only be associated with one instance of this role. This role allows the user to access and update information about any user and any provider. Essentially, this role should allow the user full access to almost anything in the system and available through the current API. Users who are assigned this role should read the ECHO Administrator's Guide as well as this document.

2.5 Transactions

There are several transactions in ECHO to facilitate the use of these roles. They are in ProviderAccountService, AdministrationService, UserAccountService, and in SessionManager.

2.5.1 SetProviderContext

If the user is associated with one or more provider roles, there must be a way to tell the system which provider they want to currently represent. Thus, each user has what is called a provider context. The provider context holds the current provider that user represents, and as the name implies, it is in the context of this provider to which the provider-oriented transactions in the system are applied.

Note: If the user is only associated with one provider role, using this transaction is not necessary. The system will assume that the user's 'provider context' holds the provider associated with that one provider role.

2.5.2 *SetUserContext*

This transaction is similar in concept to the 'SetProviderContext' transaction in ProviderAccountService. Specifically, an admin's 'user context' holds the current user to which the admin represents, and as the name implies, it is in the context of this user to which user-oriented transactions in the system are applied.

Note: Unlike the SetProviderContext transaction, there is no case where the system can assume which user the admin should represent. Thus, if an admin user does not set their 'user context' before executing a user-oriented transaction, then the transaction is applied to the actual admin user. Also there is no way to empty the 'user context' after it is set. Thus, if the admin user needs the user-oriented transactions to apply to oneself then the admin user must set the 'user context' to the actual admin user.

2.6 Queries

A Query in the ECHO system is used to search and retrieve science metadata stored by ECHO. A user can create a query by issuing a QueryRequest message of the CatalogService to ECHO. Users can search on collections (referred to as a "Discovery Search," or on granules (referred to as an "Inventory Search"). The tables below identify the kinds of queries that ECHO supports, with examples of what that search might yield.

Table 2-1: Discovery Search – Collections.

Search Criteria	Search for collections based on
Campaign	'Soil Collections','River Discharge (RIVDIS)'
Dataset ID	'15 MINUTE STREAM FLOW DATA: USGS (FIFE)'
ECHO Insert Date	The date it was inserted into ECHO; e.g. from July 3, 2001 at 5:30 pm to June 2, 2002
ECHO Last Update	The date the collection level metadata for this collection was last updated in ECHO.
Online Collections Only	Only those that are available online
Parameter	Geo-physical parameter; e.g.
Processing Level	1A, 2B, etc.
Sensor Name	'BAROMETER', 'CAMERA', 'SWIR'
Short Name	'AST_L1A', 'MYDPT1KN'
Source Name	'DIGITAL ELEVATION MODEL', 'NOAA-9'
Spatial	Polygon, polygon with holes, multi polygon
Spatial Keywords	'Global', 'United Kingdom', 'Solar'
Temporal	Date range and periodic ranges search over the temporal coverage of the collection
Temporal Keywords	'January', 'Spring', 'Stone age', 'Weekly'
PSA Names	'DAR_ID', 'QAFRACTIONGOODQUALITY','FIREPIXELS'

Table 2-2: Inventory Search – Granules.

Search Criteria	Search for collections based on
Only Granules with Browse Data	Granules that have associated browse in ECHO
Campaign	'Soil Collections','BOREAS'
Percentage of Cloud Cover	Percentage of cloud cover between 5-10%
Dataset ID	Granules from the collection identified by dataset id
ECHO Insert Date	Date it was inserted into ECHO
ECHO Last Update	granule metadata last updated in ECHO
Either Day or Night granules	Whether they are taken in day time, night time or both
Only Global Granules	Whether the granules are global
Search on granule Ids (ECHO specific)	Only search for specific granules
Search on Granule UR	'SC:AST_L1A.003:2007226177', 'VEMAP_1_SITE.w'
Online Granules Only	Only those that are available online
Path Row Range	Path row coverage; e.g., path 6, row 100
Sensor Name	'BAROMETER', 'CAMERA', 'SWIR'
Source Name	'DIGITAL ELEVATION MODEL', 'NOAA-9'
Spatial	Polygon, polygon with holes, multi polygon
Temporal	Temporal coverage of the granule
PSA	e.g. granule for DAR_ID value '345f3c'

ECHO supports storage and retrieval of previously stored queries by users. A user can also execute a previously stored query to get data that are more recent.

Currently the QueryRequest message can be used only to perform a synchronous query. This may be a drawback if the query takes too long. In future releases of ECHO, the users will be able to execute a Query asynchronously. In this context, a synchronous query is one in which the query will execute before a response is returned to the query message. An asynchronous query will receive a response to its initiating message immediately and then require an additional message to find out that the query has completed.

These future capabilities will be enabled through the CatalogService APIs that have not been currently implemented.

2.7 Results

When a query is executed in ECHO, it automatically generates a Result. Every Result in ECHO has a unique identifier within the system.

A user may execute multiple Queries simultaneously. Result sets from all these queries will be available by name (ResultSetID). These however may be removed from ECHO without notice in a reasonable amount of time, unless they were explicitly saved using the SaveResultSet transaction. Since guests are not allowed to save results in ECHO, the results are not available after the guest completes the ECHO session.

Note: Guests can not log in, so therefore guests can not log out.

Results can be retrieved by using the Present transaction to issue a PresentRequest XML message as well as through one type of query.

2.8 Catalog Items

A catalog item is any orderable item (granule or collection) in the ECHO system. Retrieving results of a query to ECHO may return several granules or collections, but not all the granules or collections may be orderable. If a granule or collection is orderable, a CatalogItemID (an XML tag in the ECHO DTD) is returned for the granule or collection metadata. Currently, for convenience of client developers, ECHO catalog item ID is in a format like "G123456-GSFC". Every catalog item ID consists of two parts. The first part starts with a capital letter, either a "G" which represents a granule item, or a "C" which represents a collection item. The numerical ID immediately follows the capital letter. The second part is the name of the provider. Two parts are linked with dash "-". To see the possible required/optional options for a catalog item, one needs to run the OrderEntryService 'PresentCatalogItem' transaction on that particular catalog item ID. Currently, there is no universal way for a provider to declare which granule or collection is orderable. Each provider must advise ECHO of their order policy or give ECHO the list of granules/collections that are orderable or searchable. Another option is for provider to follow ECHO's rule to provide orderable notification by giving the price for each orderable item even it is $0.00. Note that if a URL is provided, it is assumed that the client can simply retrieve the data from that URL as a direct link or obtain the data accessing instruction via the URL.

2.9 Orders

An order is a collection of catalog items that a user is interested in, and would presumably order. Each item in the order is associated with a quantity value, and any options available to that item. Within ECHO, a registered user creates an order and then adds, deletes, and updates each item in the order as long as any of this is done before submitting the order to ECHO. Registered users can look at the status of their orders, as well as the history of all their submitted and shipped orders.

The collection of catalog items that make up an order does not have to belong to just one provider, but can span many providers. In organizing providers and catalog items within an order, another concept called a 'provider order' is used. An order can consist of one or many provider orders. Each provider order can consist of one or many catalog items that belong to the same provider. To uniquely identify a particular provider order, one only needs the order ID that the provider order is in, and the provider ID of the provider that is associated with that provider order. When a full order is submitted to ECHO, it is these separate provider orders that are actually submitted to each associated provider.

The OrderEntryService allows registered users to create and change orders, provider orders, or individual catalog items. All transactions within the OrderEntryService deal with orders before they are submitted to the system. Once the 'Submit' transaction is executed for a certain order within the OrderEntryService, the user can no longer execute any further changes on that order. However, a user can look at the current and historical status of any of their submitted orders through the order-oriented transactions in the UserAccountService.

Once a provider order is submitted to the appropriate provider, the status of that order can be changed in two ways.

1. The provider can send an immediate response, whether they will or will not accept the order, to a provider order submission.
2. The provider can wait and asynchronously use the ProviderOrderManagementService API to change the status of an order after they have had time to fully process the order.

2.10 Groups

Groups are an aggregating mechanism in ECHO that allows a user to associate a group name with a given set of users. When a group is created, the group's owner specifies an ECHO user to be the group's manager. However, group managers can be added and removed after creation by other group managers. After becoming a member of a group, a user can be granted access to restricted metadata via the data management service.

2.11 Conditions

Conditions represent a partial equation to be evaluated as part of the ACL honoring system. The type of the condition defines the evaluation process. Temporal Conditions use a date range, and the production date of a granule is compared against the date range to check for applicability of the condition. The primary use of conditions is to facilitate reuse among data rules. The same temporal condition can be used by both a restriction and a permission to control access to metadata. To extend a time range, you only have to change one Temporal Condition as opposed to changing multiple data rules.

2.12 Rules

Rules wrap conditions and provide a complete evaluation capability as part of the ACL honoring system. Rules define which specific data is to be controlled, as well as the condition to use for evaluating if the data should be controlled. Rules also contain a comparator, which is a key part of rule evaluation. Lastly, rules contain data including ActionType (describes which actions the rule applies to), and in the case of a permission, a GroupName (describes which group the permission applies to). Restrictions (one type of a rule) apply to the global ECHO population, and Permissions (the other type of a rule) apply to a specific group.

2.13 Errors

ECHO does not currently have any Error coding system. Therefore, error messages are free text messages. An attempt is made to make the error messages as self explanatory as possible. Sometimes this means that the error messages may contain the name of the user that executed the request resulting in the error message. This makes it impossible to list all the “exact” error messages that are reported. In addition, since there is no finite set of error codes, it is hard to list down every error scenario and the corresponding error message. An attempt is made to cover different types of error messages. If you do encounter a case that is not covered here, please let us know and help us improve on this documentation.

In general, ECHO presents the error message in two major formats. On is the service responding message and another is the response from the Session Manager. Both format expressed using XML.

Error Message 1: Example of an ECHO Service responding with an error message.

```
<?xml version="1.0" standalone="no" ?>
<SubscriptionService>
   <CreateSubscriptionResponse>
      <BooleanResult>
         <BooleanResultType>
            <ERROR />
         </BooleanResultType>
         <Message>Guest users aren't allowed access to this transaction.</Message>
      </BooleanResult>
   </CreateSubscriptionResponse>
</SubscriptionService>
```

In the example above, the root TAG is the service name that you submitted you request to; the sub TAG of the service is the option response TAG with the option that you requested the service to perform. The third layer of TAG is an expression of Boolean Result, succeed or fail. If the execution of your requested option to the service failed, the ERROR is the returned result type. The Message follow give the detail indication of what is the error.

Another example (shown below) is an error responding from the Session Manager. This type of error message returned usually when the error is detected before the request reached the service layer such as invalid input XML message or the returned error message, somehow, cannot be handled by the service layer responding.

Error Message 2: Example of the ECHO Session Manager responding with an error message.

```
<?xml version="1.0" standalone="no" ?>
<SessionManager>
<Error>
   <![CDATA[
      The content of element type "SaveQueryRequest" is incomplete, it must match "(QueryName,QueryExpression)".
   ]]>
</Error>
</SessionManager>
```

Where the actual error message is listed as ![CDATA[…..]] inside the TAG of Error.

In addition to the error messages that ECHO sends out, many system level exceptions such as exceptions from java and from the Oracle database are also sent as the actual error message under certain circumstances.

3 CLIENT INTERFACE

3.1 User Registration Service

A User registers with the ECHO system by contacting the RegistrationService. The CreateUser transaction in the RegistrationService allows a guest to create a new user account and set up their profile information.

3.1.1 Transactions – CreateUser

The CreateUser transaction accepts the following fields: UserInformation, AddressInformation (optional), PhoneInformation (optional), and EmailAddress. Contained within UserInformation are fields such as: UserID, UserPassword, FirstName, LastName, and OrganizationName (optional). Currently, ECHO requires a user's password to have at least 10 characters and at least three of the following four types of characters:

1. Upper case letter
2. Lower case letter
3. Digit
4. Any other special characters such as "$", "&", ">", "<", etc.

Once registration is succeeded, UserName and Password can be used to login, and User Account Service APIs can be called.

Code Example 1: Create User.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RegistrationService PUBLIC "-//ECHO RegistrationService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/RegistrationService.dtd">
<RegistrationService>
   <CreateUserRequest>
      <UserName>NewUser1</UserName>
      <Password>1welcomeG$T</Password>
      <UserInformation>
         <FirstName>New</FirstName>
         <LastName>User</LastName>
         <EmailAddress>echodeveloper1@yahoo.com</EmailAddress>
         <OptIn>true</OptIn>
      </UserInformation>
   </CreateUserRequest>
</RegistrationService>
```

3.2 User Account Service

The User Account Service is used to update information associated with a particular user. This information includes entities such as Addresses, Emails, Phone Numbers, etc. A feature of the ECHO system is that it allows registered user to store multiple addresses, emails, and phone numbers that are distinguished by their unique names. For example, a client may set up an Address in his/her profile with the name of "Work" as well as an Address with the name of "Home". The same is true for phone numbers. The intent here is to allow a registered user to store a number of addresses and be able to choose from them. The API does not explicitly connect these, but a client interface can query for a list of addresses and fill in the blanks in the other API calls appropriately.

3.3 User Account Service Entities

3.3.1 Addresses

Address entities consist of an address id, US Format flag, a five-line street address lines, city, state, zip code, and country. A client may have multiple addresses associated with their account.

3.3.2 Emails

Only one email address can be associated with a regular user account. However, provider accounts have special capabilities beyond those of a regular client user. A Provider has multiple contacts associated with his / her account. Each contact contains an email address. Therefore, a provider account may have multiple email addresses associated with it.

3.3.3 Phone Numbers

Like Address entities, multiple phone numbers may be associated with a client user account. Phone numbers take the form of a country code, area code, exchange code, and phone.

3.3.4 Order History

Every order that is placed through the ECHO system is tracked internally. At any time, a user can ask ECHO to present his / her order history. The order history is a listing of all orders the user has transacted during their entire existence in the ECHO system; however, ECHO operations may impose time limits to effectively manage space.

3.4 Transactions

3.4.1 ListRoles

The registered user can use this transaction to list the roles associated with oneself. Currently this list would include all the provider roles and/or admin role associated with this user. The presentation of each role consists of a RoleType, a RoleTarget, and a RoleDescription. RoleType is currently either a PROVIDER or an ADMIN. RoleTarget is used in case of a PROVIDER role to show which provider this user represents. RoleDescription is for the description of the role.

Code Example 2: ListRoles request.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <ListRolesRequest/>
</UserAccountService>
```

Code Example 3: ListRoles response.

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <ListRolesResponse>
      <BooleanResult>
         <BooleanResultType>
            <REQUEST_SUCCEEDED />
         </BooleanResultType>
      </BooleanResult>
      <Role>
         <RoleType>
            <ADMIN />
         </RoleType>
         <roleDescription>Administrator of ECHO</roleDescription>
      </Role>
      <Role>
         <RoleType>
            <PROVIDER />
         </RoleType>
         <roleTarget>LPDAAC_ECS</roleTarget>
         <roleDescription>Provider Access to provider LPDAAC_ECS</roleDescription>
      </Role>
      <Role>
         <RoleType>
            <PROVIDER />
         </RoleType>
         <roleTarget>GSFCECS</roleTarget>
         <roleDescription>Provider Access to provider GSFCECS</roleDescription>
      </Role>
   </ListRolesResponse>
</UserAccountService>
```

3.4.2 PresentUserInformation

This transaction allows a user to view his or her information including: first name, last name, email address, and/or organization name.

Code Example 4: PresentUserInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentUserInformationRequest/>
</UserAccountService>
```

3.4.3 UpdateUserInformation

This transaction enables a user to update his information including: first name, last name, email address, and/or organization name.

Code Example 5: UpdateUserInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <UpdateUserInformationRequest>
      <UserInformation>
         <FirstName>NewUser</FirstName>
         <LastName>NewOne</LastName>
         <EmailAddress>NewEmail@host.com</EmailAddress>
         <OptIn>true</OptIn>
      </UserInformation>
   </UpdateUserInformationRequest>
</UserAccountService>
```

3.4.4 Present Address Information

A registered user can maintain any number of addresses. Each address will have a name that references the address; all addresses can be added, presented, updated, and deleted by this registered user. There are some address information specifics.

The name of the address (AddressID) must be unique for the user that the address belongs to, and not case sensitive.

The US format requires at least one street address line to be filled, as well as city, state, zip code and country fields.

The non-US format only requires one street address line and the country field. The country field must follow the convention of the ISO 3-letter country code.

This transaction allows a registered user to view one or more addresses in this user's profile. If AddressID is specified, the address information of the specified address is presented, or all of this user's addresses will be presented. However, presenting a non-existing address will result in an error message.

Code Example 6: PresentAddressInformation transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentAddressInformationRequest>
      <AddressID>work</AddressID>
   </PresentAddressInformationRequest>
</UserAccountService>
```

Code Example 7: PresentAddressInformation transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentAddressInformationRequest/>
```

```
</UserAccountService>
```

3.4.5 Add Address Information

This transaction adds one or more addresses to this registered user's profile. Adding an existing address will result in an error message. The AddressID identifies each address. All required address fields in an address must be filled.

Code Example 8: AddAddressInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <AddAddressInformationRequest>
      <AddressInformation>
         <AddressID>Work</AddressID>
         <USFormat>TRUE</USFormat>
         <Street1>222 bbb st.</Street1>
         <City>future city</City>
         <State>md</State>
         <Zip>22222</Zip>
         <Country>USA</Country>
      </AddressInformation>
   </AddAddressInformationRequest>
</UserAccountService>
```

3.4.6 UpdateAddressInformation

This transaction enables a registered user to update one or more addresses in this user's profile. If the user updates multiple addresses and one of them fails, then the whole update fails. Updating a new address will result in an error message. All required address fields in an address must be filled.

Code Example 9: UpdateAddressInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <UpdateAddressInformationRequest>
      <AddressInformation>
         <AddressID>work</AddressID>
         <USFormat>TRUE</USFormat>
         <Street1>9111 Edmonston Rd.</Street1>
         <Street2>Suite 202</Street2>
         <City>Greenbelt</City>
         <State>MD</State>
         <Zip>20770</Zip>
         <Country>USA</Country>
      </AddressInformation>
   </UpdateAddressInformationRequest>
</UserAccountService>
```

3.4.7 DeleteAddressInformation

This transaction enables a registered user to delete one or more addresses. If the user deletes multiple addresses and one of them could not be deleted, then the whole deletion fails. Deleting a non-existing address will result in an error message.

Code Example 10: DeleteAddressInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <DeleteAddressInformationRequest>
      <AddressID>work</AddressID>
   </DeleteAddressInformationRequest>
</UserAccountService>
```

3.4.8 PresentPhoneInformation

This transaction allows a registered user to present one or more phone numbers in his or her profile. If multiple phone numbers are to be presented and one of them cannot be presented, the whole operation fails. If PhoneName is specified, the phone information of the specified phones is presented; otherwise, all of the phones are presented. However, presenting a non-existing phone number will result in an error message.

Code Example 11: PresentPhoneInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPhoneInformationRequest>
      <PhoneName>work</PhoneName>
   </PresentPhoneInformationRequest>
</UserAccountService>
```

3.4.9 AddPhoneInformation

This transaction enables a registered user to add one or more phone numbers to a user's profile. All required phone fields must be filled, or the message is not valid. If multiple phone numbers are added and one of them cannot be added, the transaction fails. However, adding an existing phone number will result in an error message.

Code Example 12: AddPhoneInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <AddPhoneInformationRequest>
      <PhoneInformation>
         <PhoneName>work</PhoneName>
         <CountryCode>1</CountryCode>
         <AreaCode>301</AreaCode>
         <ExchangeCode>474</ExchangeCode>
         <Phone>8888</Phone>
      </PhoneInformation>
   </AddPhoneInformationRequest>
</UserAccountService>
```

3.4.10 UpdatePhoneInformation

This transaction enables a registered user to update one or more phone numbers in their profile. All required phone fields must be filled, or the message is not valid. If multiple phone numbers are to be updated and one of them cannot be updated, the transaction fails. Updating a non-existing phone number will result in an error message.

Code Example 13: UpdatePhoneInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <AddPhoneInformationRequest>
      <PhoneInformation>
         <PhoneName>work</PhoneName>
         <CountryCode>1</CountryCode>
         <AreaCode>301</AreaCode>
         <ExchangeCode>474</ExchangeCode>
         <Phone>8888</Phone>
      </PhoneInformation>
   </AddPhoneInformationRequest>
</UserAccountService>
```

3.4.11 DeletePhoneInformation

This transaction allows a registered user to delete one or more phones in his or her profile. If multiple phone numbers are to be deleted and one of them cannot be deleted, the transaction fails. PhoneNames must be specified in order to delete. Deleting non-existing phone numbers will result in an error message.

Code Example 14: DeletePhoneInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <DeletePhoneInformationRequest>
      <PhoneName>work</PhoneName>
   </DeletePhoneInformationRequest>
</UserAccountService>
```

3.4.12 PresentOptionDefinitionsForUser

Users can specify default options (or “preferences”), which ECHO uses during the ordering process. The response in this transaction lists the available options (identified in Table 3-1) that the user may set:

Table 3-1: User options.

default_shipping_address	default_contact_address
default_billing_address	default_shipping_phone

default_contact_phone	default_billing_phone
default_shipping_email	default_contact_email
default_billing_email	order_notification_level

Code Example 15: PresentOptionDefinitions transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOptionDefinitionsForUserRequest/>
</UserAccountService>
```

Note: If a particular option is identified in the request message, only information related to that option will appear in the response.

3.4.13 SetOptionSelectionsForUser

Based on the option definitions identified in the response to the PresentOptionDefinitionsForUser transaction, a user can set the options they desire.

Code Example 16: Set option selections for user.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <SetOptionSelectionsForUserRequest>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_shipping_address</OptionName>
            <Value>project</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_billing_address</OptionName>
            <Value>work</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_contact_address</OptionName>
            <Value>home</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_shipping_phone</OptionName>
            <Value>project</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
```

```
         <SimpleOptionSelection>
            <OptionName>default_billing_phone</OptionName>
            <Value>work</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_contact_phone</OptionName>
            <Value>home</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_shipping_email</OptionName>
            <Value>project@gst.com</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_billing_email</OptionName>
            <Value>work@gst.com</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_contact_email</OptionName>
            <Value>home@gst.com</Value>
         </SimpleOptionSelection>
      </OptionSelection>
   </SetOptionSelectionsForUserRequest>
</UserAccountService>
```

3.4.14 PresentOptionSelectionsForUser

This transaction enables a user to view the selections that they have set.

Code Example 17: PresentOptionSelections request.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOptionSelectionsForUserRequest>
      <OptionName>default_contact_address</OptionName>
      <OptionName>default_shipping_address</OptionName>
      <OptionName>default_billing_address</OptionName>
      <OptionName>default_contact_phone</OptionName>
      <OptionName>default_shipping_phone</OptionName>
      <OptionName>default_billing_phone</OptionName>
      <OptionName>default_contact_email</OptionName>
      <OptionName>default_shipping_email</OptionName>
      <OptionName>default_billing_email</OptionName>
   </PresentOptionSelectionsForUserRequest>
```

```
</UserAccountService>
```

Code Example 18: Response message for PresentOptionSelectionsForUserResponse.

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
  <PresentOptionSelectionsForUserResponse>
    <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED />
      </BooleanResultType>
    </BooleanResult>
    <OptionSelection>
      <SimpleOptionSelection>
       <OptionName>default_contact_address</OptionName>
       <Value>home</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_shipping_address</OptionName>
        <Value>project</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_billing_address</OptionName>
        <Value>work</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
       <OptionName>default_contact_phone</OptionName>
       <Value>home</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_shipping_phone</OptionName>
        <Value>project</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_billing_phone</OptionName>
        <Value>work</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
       <OptionName>default_contact_email</OptionName>
```

```
        <Value>home@gst.com</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_shipping_email</OptionName>
        <Value>project@gst.com</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_billing_email</OptionName>
        <Value>work@gst.com</Value>
      </SimpleOptionSelection>
    </OptionSelection>
  </PresentOptionSelectionsForUserResponse>
</UserAccountService>
```

3.4.15 Change User Password

This transaction lets the user change his or her password. Currently, ECHO requires users' passwords to have at least 10 but no more than 40 characters; the password must contain at least three of the four types of characters:

1. Upper case letter
2. Lower case letter
3. Digit
4. Any other special characters such as "$", "&", ">", "<", etc.

A regular user must provide the old password for security reasons.

Note: An admin user may change any other user's password without providing the old password. However, if the admin user changes his or her own password, the old password is still required.

Code Example 19: ChangeUserPassword transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <ChangeUserPasswordRequest>
      <OldPassword>1welcomeG$T</OldPassword>
      <NewPassword>newPassword3</NewPassword>
   </ChangeUserPasswordRequest>
</UserAccountService>
```

3.4.16 PresentOrderHistory

The order APIs in the User Account Service are used to track a users order history, pending orders and cancelled orders. These APIs should be post submitted transactions. Only registered users can use these APIs since they are in User Account Service.

This transaction enables the registered user to view detailed information about user's orders that have been fulfilled, or are in terminating order states, in which orders are no longer active. The terminating states include SUBMITTED_WITH_EXCEPTIONS, CANCELLED, CLOSED, CLOSED_WITH_EXCEPTIONS.

Code Example 20: PresentOrderHistory transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOrderHistoryRequest/>
</UserAccountService>
```

Code Example 21: PresentOrderHistory transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOrderHistoryRequest>
      <TerminatingOrderState>
         <CLOSED/>
      </TerminatingOrderState>
      <TerminatingOrderState>
         <CLOSED_WITH_EXCEPTIONS/>
      </TerminatingOrderState>
   </PresentOrderHistoryRequest>
</UserAccountService>
```

3.4.17 *PresentOrderHistorySummary*

This transaction is similar to present order history, except it just returns order ID and order state.

Code Example 22: PresentOrderHistorySummary transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOrderHistorySummaryRequest/>
</UserAccountService>
```

Code Example 23: PresentOrderHistorySummary transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOrderHistorySummaryRequest>
      <TerminatingOrderState>
         <CLOSED/>
      </TerminatingOrderState>
      <TerminatingOrderState>
         <CLOSED_WITH_EXCEPTIONS/>
      </TerminatingOrderState>
   </PresentOrderHistorySummaryRequest>
</UserAccountService>
```

3.4.18 *PresentPendingOrders*

This transaction enables the registered user to view detailed information about the user's orders that have not been fulfilled, or are in pending order states. The pending order states are SUBMITTING, PROCESSING, PROCESSING_WITH_EXCEPTIONS, CANCELLING.

Code Example 24: PresentPendingOrder transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPendingOrdersRequest/>
</UserAccountService>
```

Code Example 25: PresentPendingOrder transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPendingOrdersRequest>
      <PendingOrderState>
         <SUBMITTING/>
      </PendingOrderState>
      <PendingOrderState>
         <PROCESSING/>
      </PendingOrderState>
   </PresentPendingOrdersRequest>
</UserAccountService>
```

3.4.19 *PresentPendingOrderSummary*

This transaction is similar to present pending order, except it just returns order ID and order state.

Code Example 26: PresentPendingOrderSummary transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPendingOrderSummaryRequest/>
</UserAccountService>
```

Code Example 27: PresentPendingOrderSummary transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentPendingOrderSummaryRequest>
      <PendingOrderState>
         <SUBMITTING/>
      </PendingOrderState>
      <PendingOrderState>
         <PROCESSING/>
      </PendingOrderState>
   </PresentPendingOrderSummaryRequest>
</UserAccountService>
```

3.4.20 CancelOrder

The registered user may request cancellation of an order or a provider order that has been submitted and has not yet been fulfilled. There may be a cost associated with canceling the order. Either a whole order can be specified (by the first OrderID) or a specific provider order can be specified (by the ProviderOrderID). Specifying both OrderID and ProviderOrderID will result in an error message .

Code Example 28: CancelOrder transaction (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <CancelOrderRequest>
      <ProviderOrderID>
         <ProviderID>1111:0000000</ProviderID>
         <OrderID>0000000</OrderID>
      </ProviderOrderID>
   </CancelOrderRequest>
</UserAccountService>
```

Code Example 29: CancelOrder transaction (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <CancelOrderRequest>
      <OrderID>0000000</OrderID>
   </CancelOrderRequest>
</UserAccountService>
```

3.5 Error Messages

Table 3-2 gives the error messages associated with each transaction in the User Account Service.

Table 3-2: User Account Service error messages.

Transaction	Error Message	Description
Add Address Information	This address already exists. Please add a new one	This error is returned when the address already exists.
Update Address Information	This address does not exist.	This error is returned if the address indicated for update does not exist.
Present Address Information	<addressID> is not found.	This error is returned when trying to request for an address that does not exist.
Delete Address Information	<addressID> is not found.	This error is returned when trying to delete an address that does not exist.
Add Phone Information	This phone number already exists. Please add a new one	This error is returned when trying to add a phone using a name that already exists.
Update Phone Information	This phone number does not exist.	This error is returned when trying to update a phone that does not exist.
Present Phone Information	EJB Exception: : java.lang.NullPointerException at	This error is returned when requesting for a phone using phone name that does not exist.
Delete Phone Information	<phoneName> is not found.	This error is returned when trying to delete a phone that does not exist.
Change User Password	The old password is incorrect.	This error is returned when entering an incorrect current password for password changing.
Cancel Order	Unable to locate order <orderID>.	This error is returned when trying to cancel an order that does not exist for this user.
	The provider <ProviderID> does not exist in the order <OrderID>.	This error is returned when the provider ID indicated is not included in the order referred by the order ID.
	Cancellation of the order from provider:<ProviderID> has already been rejected by the provider.	This error is returned when trying to again cancel an order that cancellation on this order was rejected by the provider once before.
	Provider Order from <ProviderID> is in an invalid state to be cancelled.	This error is returned when user trying to cancel an order that is placed in a state that cannot be cancelled.
Set Option Selections	[<OptionName>] is not a valid property for the item.	This error is returned when trying to access an option that is not valid for the item.

Transaction	Error Message	Description
	There is no property <ChildOption> defined for the type: <ParentOption>	This error is returned when submitting an invalid option structure.

3.6 Catalog Service

This service allows users to query ECHO's metadata clearinghouse that contains catalog items such as data sets (“collections”) and granules. ECHO allows users to search on

1. Campaign or project (e.g., TRMM, IAA Environmental Program)
2. Data set
3. Date
4. Sensor (e.g., Acoustic Sounders, CO2 Analyzers)
5. Source or platform (e.g., Aircraft, TERRA, Apollo, Gravity Stations)
6. Geographic area
7. Time periods

This service supports the storage and execution of named queries and the storage of named results. The query is written in the IIMS Alternative Query Language (AQL).

3.7 Transactions

3.7.1 ExplainCollection

This transaction is not yet implemented.

3.7.2 ExplainSearchParameter

This transaction is used to retrieve valid values for a particular IIMSAQL search criteria.

A query to ECHO can contain multiple search criteria. Only those collections/granule that satisfy all the search criteria are returned. If the user does not have prior knowledge (which is the usual case) of the metadata holdings of ECHO, it may happen that the combination of search criteria specified in the query, return 0 hits. If there was a way to provide the user with information about the valid values for search criteria, it would reduce the occurrence of 0 hit queries.

An ExplainSearchParameter request takes in 2 elements:

1. SearchParameterName specifies the IIMSAQL search criteria name for which the valid values must be returned.
2. QueryExpression (optional) that may be used to further limit the valid values returned based on other search criteria that the user has selected.

This feature is particularly useful for dynamically building the query. For example, suppose the user is only interested in online granules for ORNL provider. Further, he or she would like the best quality data in terms of cloud cover. It may be useful to know, given the previous requirements, what are the valid values for cloud cover. Therefore, if the user knows the best data still has 9% cloud cover, he or she can make sure not to query for data with only 5% cloud cover that would result in a 0 hit query.

If the QueryExpression element is not specified, it will return valid values for the search criteria for all the metadata in the ECHO system. Two things govern the content of the response message that is returned:

1. The type of valid values for the search criteria: For example, cloudCover is a percentage that can take a range of values, so the response message contains a CriteriaRange element. A response message of CampaignShortName in turn would contain CriteriaValues.

2. Whether there are any valid values for the search criteria, given (if any) other restrictions: If no valid values exist in ECHO for a particular search criteria, or given a particular restriction on the valid values (by specifying the QueryExpression in the request) results in invalid values, the returned Criteria element will only contain CriteriaName, CriteriaType (and CriteriaLength if the criteria takes String values). Neither CriteriaRange nor CriteriaValues will be returned.

Valids for a few search criteria in IIMSAQL are currently not available. The unavailable criteria are temporal, ECHOInsertDate, ECHOLastUpdate, spatial, pathRow. In addition since onlineOnly, browseOnly, dayNightFlag are function based search criteria and do not have valid values, these are also not supported through ExplainSearchParameter.

Code Example 30: ExplainSearchParameter request.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <ExplainSearchParameterRequest>
      <SearchParameterName>cloudCover</SearchParameterName>
      <QueryExpression>
        <query><![CDATA[

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN" "http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd
">
<query>
   <for value="granules"/>
   <dataCenterId>
       <value>LPDAAC_ECS</value>
   </dataCenterId>
   <where>
      <granuleCondition>
         <onlineOnly/>
      </granuleCondition>
   </where>
</query>

]]></query>
      <namespace>none</namespace>
         <QueryLanguage>
            <IIMSAQL/>
         </QueryLanguage>
      </QueryExpression>
   </ExplainSearchParameterRequest>
</CatalogService>
```

Code Example 31: ExplainSearchParameter response.

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <ExplainSearchParameterResponse>
      <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED />
      </BooleanResultType>
    </BooleanResult>
    <Valid>
      <Category>
        <CategoryName>cloudCover</CategoryName>
        <CategoryType>GRANULES</CategoryType>
        <CategoryDescription>Scene Cloud Coverage in percentage</CategoryDescription>
        <NegationAllowed>N</NegationAllowed>
          <CriteriaList>
            <Criteria>
              <CriteriaName>cloudCover</CriteriaName>
              <CriteriaType>float</CriteriaType>
              <CriteriaRange>
                <RangeMin>0</RangeMin>
                <RangeMax>100</RangeMax>
              </CriteriaRange>
            </Criteria>
          </CriteriaList>
      </Category>
    </Valid>
  </ExplainSearchParameterResponse>
</CatalogService>
```

3.7.3 *ListSavedQueries*

This message is used to request a list of the names of the saved queries. Guests do not have access to this transaction.

3.7.4 *ListSavedResultSets*

This message is used to request a list of the names of the saved result sets. Guests do not have access to this transaction.

Code Example 32: List Saved ResultSets transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<!-- Description:  Lists saved result sets. -->
<CatalogService>
  <ListSavedResultSetsRequest/>
</CatalogService>
```

3.7.5 *Present*

To retrieve the results of a previously executed query to ECHO, the Present transaction of the CatalogService should be used to send a PresentRequest XML message to ECHO. This message mainly consists of the ResultSetID (that is returned as part of the response to the previously executed query) and results presentation specification.

The following is a sample PresentRequest message to return results from a search for granules. Note that the ResultSetID value should match the value you get from the QueryResponse message.

Code Example 33: Present API call.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <PresentRequest>
      <ResultSetID>RU10021060199867864</ResultSetID>
      <PresentationDescription>
         <TupleType>
            <attributeName>GranuleUR</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>GranuleVersionId</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>sizeMBDataGranule</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>price</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>ECHOItemId</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>

         <TupleType>
           <attributeName>RangeBeginningDate</attributeName>
           <PrimitiveTypeName>
              <String/>
           </PrimitiveTypeName>
         </TupleType>

         <TupleType>
            <attributeName>ShortName</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>DayNightFlag</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>

         <TupleType>
            <attributeName>ECHOInsertDate</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>

         <TupleType>
            <attributeName>Sensor</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>

         <DTDType>
            <ECHO/>
         </DTDType>
      </PresentationDescription>
      <IteratorSize>5</IteratorSize>
      <Cursor>1</Cursor>
   </PresentRequest>
</CatalogService>
```

ResultSetID specifies the result set that should be presented. If it is not specified it defaults to the last executed query-result. For registered users this could be from a previous session. For guests it must be from the same session. This is a mandatory field.

3.7.5.1 Query Results Presentation Specification

The result presentation specification can be used to specify the subset of the information that should be returned as results of a query. The description below is applicable to both PresentRequest message and QueryRequest (when requesting results as part of QueryResponse message).

The following elements are used to specify the format and content of the results of a query:

1. **MessageFormat.** Specifies the format of the results. It takes values XML, HTML , TXT. Currently ECHO only supports XML.
2. **Cursor.** Specifies the first record to be returned. e.g. a value of 5 will return results starting from the 5th record. If none is specified it defaults to 1.
3. **IteratorSize.** Specifies the number of results to be returned. e.g. a value of 10 combined with the Cursor value will return results 5-14. In one transaction at most 2000 results will be returned.
4. **PresentationDescription.** Defines the format of the results. It is divided into 3 parts. DTDType and TupleTypes and PredefinedPresentationType. The DTDType defines the structure of the results. In case of XML MessageFormat, DTDType is used to specify the XML results are to be returned in accordance to what DTD.

The other elements SortField, QueryScope, CollectionName, and CatalogEntryType have been added for possible future requirements. They are not currently implemented.

3.7.5.2 PresentationDescription

The two parts in the PresentationDescription together determine the XML that is returned. As mentioned above the DTDType determines the DTD the results will conform to.

The DTDs for ECHO and BMGT closely match the ECHO ingest and BMGT ingest DTDs but in addition, they also contain some ECHO generated information like ECHOInsertDate etc. The DTDs for granules and collections are separate to eliminate any inconsistencies due to overlapping elements with different definitions. The following two tables list the granule and collection DTDs for each DTDType.

Table 3-3: DTDs for Granules.

ECHO	http://api.echo.eos.nasa.gov/echo/dtd/ECHOGranuleResults.dtd
BMGT	http://api.echo.eos.nasa.gov/echo/dtd/BMGTGranuleResults.dtd

Table 3-4: DTDs for Collections.

ECHO	http://api.echo.eos.nasa.gov/echo/dtd/ECHOCollectionResults.dtd
BMGT	http://api.echo.eos.nasa.gov/echo/dtd/BMGTCollectionResults.dtd

DTDType defaults to <ECHO/>. If the TupleType elements are not specified, all the elements that can be returned for the specified DTDType will be returned. This may be time and resource consuming. If not all the elements are required, then a subset of these elements can be specified in the TupleTypes and only those will be returned.

TupleType takes the name of the result attribute (element) in the attributeName element. It also has an element PrimitiveTypeName that specifies the data type of the attribute. Currently, PrimitiveTypeName is ignored.

The list of possible result attributes (elements) that can be specified in TupleTypes depends on the DTD to which the results will conform. For example, if the results conform to http://api.echo.eos.nasa.gov/echo/dtd/ECHOGranuleResults.dtd, then the elements that can be specified are only all XML elements defined in this DTD under and including GranuleURMetaData.

Whatever element is specified in the TupleType, all the elements under it will also be returned in the results. For example, if Platform is specified, Platform, PlatformShortName, Instrument, InstrumentShortName, Sensor, SensorShortName, SensorCharacteristitcs, SensorCharacteristicName, SensorCharacteristicValue and OperationMode, will also be returned.

If only Sensor is specified, then all the key elements above it – InstrumentShortName, Instrument, PlatformShortName and Platform, GranuleUR and GranuleURMetaData – will also be returned. This is to ensure that the data is identified correctly.

Please note for spatial information BoundingBox, Polygon etc., CANNOT be used in the TupleType. Only the element enclosing the spatial bounding rectangle/polygon can be specified in the TupleType. For example, for ECHO format results for Granules, to get spatial information HorizontalSpatialDomainContainer should be specified.

The spatial elements identified in Table 3-5 cannot be specified as TupleTypes:

Table 3-5: Spatial elements that cannot be specified as TupleTypes.

Point	Circle	BoundingRectangle
GPolygon	Polygon	PointLongitude
PointLatitude	CenterLatitude	CenterLongitude
Radius	WestBoundingCoordinate	NorthBoundingCoordinate
EastBoundingCoordinate	SouthBoundingCoordinate	Boundary
ExclusiveZone	SinglePolygon	MutiPolygon
OutRing	InnerRing	

Specifying GranuleURMetaData as a TupleType to be returned is equivalent to not specifying any TupleTypes, as the result is the same as returning all the elements in the result DTD.

PredefinedPresentationType is not currently implemented. It will be used in the future to specify an ECHO defined name that represents a predetermined set of TupleTypes. Currently this field is ignored.

If PresentationDescription is not specified at all, it defaults to DTDType to be <ECHO/> and no TupleTypes.

An example QueryRequest with nothing specified for PresentationDescription (taking default values) and its response with results follows. Note that the actual result string is embedded in a wrapper XML message within a CDATA field, so that even though it looks like XML, it will have to be extracted in order to parse it as XML—This is done to accommodate non-XML return formats.

Code Example 34: QueryRequest with default PresentationDescription.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
    <QueryRequest>
        <QueryExpression>
            <query>
<![CDATA[

<?xml version="1.0" encoding="UTF-8"?>


<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
    <for value="granules"/>
        <dataCenterId>
            <value>ORNL-DAAC</value>
        </dataCenterId>
    <where>
        <granuleCondition>
            <dataSetId><textPattern>'%SAT%'</textPattern></dataSetId>
        </granuleCondition>
    </where>
</query>

]]>
```

```
            </query>
            <namespace>none</namespace>
            <QueryLanguage>
                <IIMSAQL/>
            </QueryLanguage>
        </QueryExpression>
        <ResultType>
            <RESULTS/>
        </ResultType>
        <IteratorSize>2</IteratorSize>
    </QueryRequest>
</CatalogService>
```

Code Example 35: Query response.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
    <QueryResponse>
        <BooleanResult>
            <BooleanResultType>
                <REQUEST_SUCCEEDED/>
            </BooleanResultType>
        </BooleanResult>
        <ReturnData>
        <MessageFormat>
            <XML/>
        </MessageFormat>
        <payload>
            <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "-//ECHO results (v5.5)//EN" "http://api.echo.eos.nasa.gov/echo/dtd/ECHOGranuleResults.dtd">
<results>
    <provider name='ORNL-DAAC'>
        <result itemId='G712283-ORNL' number='1'>
            <GranuleURMetaData>
                <ECHOItemId>G71283-ORNL-DAAC</ECHOItemId>
                <GranuleUR>BOREAS_RADARSAT.data_readme</GranuleUR>
                <ECHOInsertDate>2001-05-15 16:59:37.0</ECHOInsertDate>
                <ECHOLastUpdate>2001-05-15 16:59:46.0</ECHOLastUpdate>
                <GranuleShortName>data_readme</GranuleShortName>
                <AccessConstraint>PUBLIC</AccessConstraint>
                <CollectionMetaData>
                    <ShortName>BOREAS RADARSAT IMAGES CD-ROM</ShortName>
                    <VersionID>0</VersionID>
                    <DataSetId>BOREAS RADARSAT IMAGES CD-ROM</DataSetId>
                </CollectionMetaData>
                <DataGranule>
                    <SizeMBDataGranule>66</SizeMBDataGranule>
                </DataGranule>
                <SpatialDomainContainer>
                    <HorizontalSpatialDomainContainer>
                        <BoundingRectangle>
                            <WestBoundingCoordinate>-111</WestBoundingCoordinate>
                            <NorthBoundingCoordinate>60</NorthBoundingCoordinate>
                            <EastBoundingCoordinate>-93</EastBoundingCoordinate>
                            <SouthBoundingCoordinate>49</SouthBoundingCoordinate>
                        </BoundingRectangle>
                    </HorizontalSpatialDomainContainer>
                </SpatialDomainContainer>
                <MeasuredParameter>
                    <MeasuredParameterContainer>
                        <ParameterName>RADAR BACKSCATTER</ParameterName>
                    </MeasuredParameterContainer>
                </MeasuredParameter>
                <StorageMediumClass>
                    <StorageMedium>On-Line</StorageMedium>
                </StorageMediumClass>
                <Platform>
                    <PlatformShortName>RADARSAT-1</PlatformShortName>
                    <Instrument>
                        <InstrumentShortName>SAR</InstrumentShortName>
                        <Sensor>
                            <SensorShortName>SAR</SensorShortName>
                        </Sensor>
                    </Instrument>
                </Platform>
                <Campaign>
                    <CampaignShortName>BOREAS</CampaignShortName>
                </Campaign>
                <Contact>
                    <Role>User Services Office</Role>
                    <ContactOrganizationName>ORNL DAAC User Services Office</ContactOrganizationName>
                    <ContactOrganizationAddress>
                        <StreetAddress>Oak Ridge National Laboratory,      P.O. Box 2008, MS 6407</StreetAddress>
                        <City>Oak Ridge</City>
                        <StateProvince>Tennessee</StateProvince>
                        <PostalCode>37831-6407</PostalCode>
                        <Country>USA</Country>
                    </ContactOrganizationAddress>
                    <OrganizationTelephone>
                        <TelephoneNumber>+1 (865) 241-3952</TelephoneNumber>
                        <TelephoneType>Telephone</TelephoneType>
                    </OrganizationTelephone>
                    <OrganizationEmail>
                        <ElectronicMailAddress>ornldaac@ornl.gov</ElectronicMailAddress>
                    </OrganizationEmail>
                </Contact>
                <OnlineAccessURL>http://daac.ornl.gov/boreas/GRAB_BAG/radarsat/data/data_readme</OnlineAccessURL>
            </GranuleURMetaData>
        </result>

        <result number='2' itemId='G108456-ORNL-DAAC'>
            <GranuleURMetaData>
```

```
<ECHOItemId> G108456-ORNL-DAAC</ECHOItemId>
<GranuleUR>FIFE_SAT_AVHR.7034fife.avh</GranuleUR>
<ECHOInsertDate>2001-05-15 22:51:57.0</ECHOInsertDate>
<ECHOLastUpdate>2001-05-15 22:52:05.0</ECHOLastUpdate>
<GranuleShortName>7034fife.avh</GranuleShortName>
<AccessConstraint>PUBLIC</AccessConstraint>
<CollectionMetaData>
     <ShortName>SATELLITE AVHRR EXTRACTED DATA (FIFE)</ShortName>
     <VersionID>0</VersionID>
     <DataSetId>SATELLITE AVHRR EXTRACTED DATA (FIFE)</DataSetId>
</CollectionMetaData>
<DataGranule>
     <SizeMBDataGranule>10000</SizeMBDataGranule>
</DataGranule>
<SpatialDomainContainer>
     <HorizontalSpatialDomainContainer>
          <BoundingRectangle>
               <WestBoundingCoordinate>-96.625</WestBoundingCoordinate>
               <NorthBoundingCoordinate>39.125</NorthBoundingCoordinate>
               <EastBoundingCoordinate>-96.625</EastBoundingCoordinate>
               <SouthBoundingCoordinate>39.125</SouthBoundingCoordinate>
          </BoundingRectangle>
     </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
<MeasuredParameter>
     <MeasuredParameterContainer>
          <ParameterName>IRRADIANCE REFLECTANCE</ParameterName>
     </MeasuredParameterContainer>
</MeasuredParameter>
<StorageMediumClass>
     <StorageMedium>On-Line</StorageMedium>
</StorageMediumClass>
<Platform>
     <PlatformShortName>NOAA-9 NOAA-10 NOAA-11</PlatformShortName>
     <Instrument>
          <InstrumentShortName>AVHRR</InstrumentShortName>
          <Sensor>
               <SensorShortName>AVHRR</SensorShortName>
          </Sensor>
     </Instrument>
</Platform>
<Campaign>
     <CampaignShortName>FIFE</CampaignShortName>
</Campaign>
     <Contact>
          <Role>User Services Office</Role>
          <ContactOrganizationName>ORNL DAAC User Services Office</ContactOrganizationName>
          <ContactOrganizationAddress>
               <StreetAddress>Oak Ridge National Laboratory,      P.O. Box 2008, MS
6407</StreetAddress>
               <City>Oak Ridge</City>
               <StateProvince>Tennessee</StateProvince>
               <PostalCode>37831-6407</PostalCode>
               <Country>USA</Country>
```

```
                              </ContactOrganizationAddress>
                              <OrganizationTelephone>
                                   <TelephoneNumber>+1 (865) 241-3952</TelephoneNumber>
                                   <TelephoneType>Telephone</TelephoneType>
                              </OrganizationTelephone>
                              <OrganizationEmail>
                                   <ElectronicMailAddress>ornldaac@ornl.gov</ElectronicMailAddress>
                              </OrganizationEmail>
                         </Contact>
                    <OnlineAccessURL>http://daac.ornl.gov/fife_1/data/sat_obs/sat_avhr/y1987/7034fife.avh</OnlineAccessURL>
               </GranuleURMetaData>
          </result>
     </provider>
</results>]]>
          </payload>
          </ReturnData>
          <RequestID>RGuest7213396921014333628487</RequestID>
          <ResultSetID>RGuest7213396921014333628487</ResultSetID>
          <ResultType>
               <RESULTS/>
          </ResultType>
          <Status>
               <SUCCESS_RESULTS_AVAILABLE/>
          </Status>
         <Hits>85</Hits>
         <Cursor>3</Cursor>
     </QueryResponse>
</CatalogService>
```

The DTD for the result (as noted in the XML itself) is located at:

http://api.echo.eos.nasa.gov/echo/dtd/ECHOGranuleResults.dtd

The root element, <results> begins every result XML document. The results are grouped per provider. Hence the element under result is provider with an attribute name that specifies the name of the provider.

3.7.6 GetMetadata

Users have flexibility of getting metadata information without providing resultSetID that was obtained from previous Query API call. Instead, by specifying the ECHOItemIDs they are interested in directly, users can get the metadata information through GetMetadata API call. The parameters that are required to perform the GetMetadata are ItemId, as well as MessageFormat and PresentationDescription, which have been described in the Present API section. One restriction in using this transaction is that users have to make sure all the item ids must have the same data type, either COLLECION or GRANULE, otherwise an exception will be thrown to state which item id has a problem with the data type.

Code Example 36: GetMetadata API call.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <GetMetadataRequest>
      <ItemID>C10800-ORNL-DAAC</ItemID>
      <ItemID>C10802-ORNL-DAAC</ItemID>
      <PresentationDescription>
         <TupleType>
            <attributeName>Sensor</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>Campaign</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <DTDType>
            <BMGT/>
         </DTDType>
      </PresentationDescription>
   </GetMetadataRequest>
</CatalogService>
```

Code Example 37: Response message. Note that the message follows the first version of BMGT.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <GetMetadataResponse>
      <BooleanResult>
         <BooleanResultType>
            <REQUEST_SUCCEEDED/>
         </BooleanResultType>
      </BooleanResult>
      <ReturnData>
         <MessageFormat>
            <XML/>
         </MessageFormat>
         <payload>
            <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "-//ECHO results (v5.5)//EN" "http://api.echo.eos.nasa.gov/echo/dtd/BMGTCollectionResults.dtd">
<results>
   <provider name='ORNL-DAAC'>
    <result itemId='C10800-ORNL-DAAC'>
      <CollectionMetaData>
        <ECHOItemId>C10800-ORNL-DAAC</ECHOItemId>
        <DataSetId>BOREAS AES FIVE-DAY AVERAGED SURFACE METEOROLOGICAL AND UPPER AIR DATA</DataSetId>
        <Platform>
          <PlatformShortName>METEOROLOGICAL STATION</PlatformShortName>
          <Instrument>
            <InstrumentShortName>HUMAN OBSERVER</InstrumentShortName>
            <Sensor>
              <SensorShortName>HUMAN OBSERVER</SensorShortName>
              <SensorLongName>HUMAN OBSERVER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>ANEMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>ANEMOMETER</SensorShortName>
              <SensorLongName>ANEMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>BAROMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>BAROMETER</SensorShortName>
              <SensorLongName>BAROMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>DEWPOINT HYDROMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>DEWPOINT HYDROMETER</SensorShortName>
              <SensorLongName>DEWPOINT HYDROMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>DRY BULB THERMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>DRY BULB THERMOMETER</SensorShortName>
              <SensorLongName>DRY BULB THERMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>RAIN GAUGE</InstrumentShortName>
            <Sensor>
              <SensorShortName>RAIN GAUGE</SensorShortName>
              <SensorLongName>RAIN GAUGE</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>SNOW MEASURING ROD</InstrumentShortName>
            <Sensor>
              <SensorShortName>SNOW MEASURING ROD</SensorShortName>
              <SensorLongName>SNOW MEASURING ROD</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>WET BULB THERMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>WET BULB THERMOMETER</SensorShortName>
              <SensorLongName>WET BULB THERMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
        </Platform>
        <Campaign>
```

```
        <CampaignShortName>BOREAS</CampaignShortName>
        <CampaignLongName>BOREAS</CampaignLongName>
      </Campaign>
    </CollectionMetaData>
  </result>
  <result itemId='C10802-ORNL-DAAC'>
    <CollectionMetaData>
      <ECHOItemId>C10802-ORNL-DAAC</ECHOItemId>
      <DataSetId>BOREAS AES CANADIAN HOURLY AND DAILY SURFACE METEOROLOGICAL DATA</DataSetId>
      <Platform>
        <PlatformShortName>METEOROLOGICAL STATION</PlatformShortName>
        <Instrument>
          <InstrumentShortName>HUMAN OBSERVER</InstrumentShortName>
          <Sensor>
            <SensorShortName>HUMAN OBSERVER</SensorShortName>
            <SensorLongName>HUMAN OBSERVER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>ANEMOMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>ANEMOMETER</SensorShortName>
            <SensorLongName>ANEMOMETER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>BAROMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>BAROMETER</SensorShortName>
            <SensorLongName>BAROMETER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>DEWPOINT HYDROMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>DEWPOINT HYDROMETER</SensorShortName>
            <SensorLongName>DEWPOINT HYDROMETER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>DRY BULB THERMOMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>DRY BULB THERMOMETER</SensorShortName>
            <SensorLongName>DRY BULB THERMOMETER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>RAIN GAUGE</InstrumentShortName>
          <Sensor>
            <SensorShortName>RAIN GAUGE</SensorShortName>
            <SensorLongName>RAIN GAUGE</SensorLongName>
          </Sensor>
        </Instrument>
```

```
          <Instrument>
            <InstrumentShortName>SNOW MEASURING ROD</InstrumentShortName>
            <Sensor>
              <SensorShortName>SNOW MEASURING ROD</SensorShortName>
              <SensorLongName>SNOW MEASURING ROD</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>WET BULB THERMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>WET BULB THERMOMETER</SensorShortName>
              <SensorLongName>WET BULB THERMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
        </Platform>
        <Campaign>
          <CampaignShortName>BOREAS</CampaignShortName>
          <CampaignLongName>BOREAS</CampaignLongName>
        </Campaign>
      </CollectionMetaData>
    </result>
  </provider>
</results>]]>
      </payload>
    </ReturnData>
  </GetMetadataResponse>
</CatalogService>
```

3.7.7 PresentSavedQuery

This message is used to return the details of a query that has been saved. Guests do not have access to this transaction.

Code Example 38: PresentSavedQuery transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<!-- Description:  Lists saved result sets. -->
<CatalogService>
  <PresentSavedQueryRequest>
   <QueryName>mq1</QueryName>
</PresentSavedQueryRequest>
</CatalogService>
```

3.7.8 Query

To query ECHO for provider metadata the Query transaction of the CatalogService should be used to send a QueryRequest message to ECHO. This message mainly consists of the query and result presentation specification.

Code Example 39: Query Request transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <QueryRequest>
     <QueryExpression>
        <query>
   <![CDATA[
      …IIMSAQL query goes here…
   ]]>
        </query>
        <namespace>none</namespace>
        <QueryLanguage>
            <IIMSAQL/>
        </QueryLanguage>
    </QueryExpression>
    <ResultType>
       <RESULTS/>
    </ResultType>
    <IteratorSize>10</IteratorSize>
    <Cursor>1</Cursor>
    <PresentationDescription>
       <TupleType>
           <attributeName>Platform</attributeName>
           <PrimitiveTypeName><Integer/></PrimitiveTypeName>
       </TupleType>
       <TupleType>
           <attributeName>SensorShortName</attributeName>
           <PrimitiveTypeName><String/></PrimitiveTypeName>
       </TupleType>
       ... more TupleType specification ...
       <DTDType>
           <ECHO/>
       </DTDType>
    </PresentationDescription>
 </QueryRequest>
</CatalogService>
```

The query must be specified as text under the <query> element in IIMSAQL query language and must conform to the IIMSAQLQueryLanguage.dtd in accordance with the IIMSAQL specification in section on Alternative Query Language.

Note: The actual query (in IIMSAQL) is enclosed in CDATA section in the QueryRequest message and hence a parser that is validating the QueryRequest message will not validate it. The query enclosed in the CDATA section should first be validated by itself against the IIMSAQLQueryLanguage.dtd before inserting it into the QueryRequest message.

The presentation portion is split into 2 areas. The first is related to the kind of response required to the QueryRequest message. This is specified by the ReturnType element that takes the values: RESULTS, RESULT_SET_ID, HITS, ITEM_IDS. It also takes the value VALIDATE but it is not currently implemented and may be deprecated in the future.

RESULTS. Implies that the results should be returned as part of the response to the query request. When using this option, the rest of the details regarding presentation must be specified in the QueryRequest message. In addition, a key (ResultSetID) is also returned for subsequent retrievals of results in case all the results were not returned by the query response.

RESULT_SET_ID. Implies that the response should only return a key (ResultSetID) and the results will be subsequently retrieved using the Present transaction. No results are returned.

HITS. Is similar to RESULT_SET_ID wherein the number of hits is returned in addition to RESULT_SET_ID. No results are returned. The implication is that a count of results has to be performed and may take slightly longer to return.

ITEM_IDS. Implies that the matched item ids should be returned to the user directly. Also the total number of item ids is returned. (Note: No result set ID is returned since results are not persisted in the system nor are they used to provide a stateless version of the Query transaction.) This implies that no RESULT is created within ECHO to store the results of the query. All the Ids of the granules/collections that satisfy the query are returned to the client. It is the client's responsibility to request the metadata for each individual granule/collection using the GetMetadata transaction discussed later.

The second area is applicable for QueryRequest message only if the results (part or all) are to be returned as part of the response to the QueryRequest message (by setting the ReturnType to RESULTS). This area is also used in the PresentRequest message; it is discussed in the Result Specification Section.

Code Example 40: Query Request message from "BOREAS" campaign.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<!-- Description:    Query ORNL data for granules from 'BOREAS' campaign. -->
<CatalogService>
   <QueryRequest>
      <QueryExpression>
         <query>
            <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!-- Search ORNL for granules from BOREAS campaign -->
<query>
   <for value="granules"/>
      <dataCenterId>
         <value>ORNL-DAAC</value>
      </dataCenterId>
   <where>
      <granuleCondition>
         <CampaignShortName><value>'Boreas'</value></CampaignShortName>
      </granuleCondition>
   </where>
</query>
]]>
         </query>
         <namespace>none</namespace>
         <QueryLanguage>
           <IIMSAQL/>
         </QueryLanguage>
      </QueryExpression>
      <ResultType>
         <RESULT_SET_ID/>
      </ResultType>
   </QueryRequest>
</CatalogService>
```

3.7.9 RemoveSavedQuery

This message is used to remove a saved query. This transaction is not available to Guests.

Code Example 41: RemoveSavedQuery transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
      <RemoveSavedQueryRequest>
         <QueryName>mq1</QueryName>
      </RemoveSavedQueryRequest>
</CatalogService>
```

3.7.10 RemoveSavedResultSet

This message is used to remove a saved result. This transaction is not available to Guests.

Code Example 42: RemoveSavedResultSet transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <RemoveSavedResultSetRequest>
      <ResultSetID>result1</ResultSetID>
   </RemoveSavedResultSetRequest>
</CatalogService>
```

3.7.11 SaveQuery

This is a request to save a query. Once a query has been saved, it may be executed by name rather than providing the entire query expression.

Code Example 43: SaveQuery transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <SaveQueryRequest>
   <QueryName>mq1</QueryName>
   <QueryExpression>
      <query>
      <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
 <for value="granules"/>
  <dataCenterId>
     <all/>
  </dataCenterId>
 <where>
  <granuleCondition>
     <onlineOnly/>
  </granuleCondition>
 </where>
</query>
]]>
      </query>
      <namespace>none</namespace>
      <QueryLanguage>
         <IIMSAQL></IIMSAQL>
      </QueryLanguage>
   </QueryExpression>
</SaveQueryRequest>
</CatalogService>
```

3.7.12 SaveResultSet

This message is used to persist a Result Set so that it may be retrieved or searched at a later time. This transaction is not available to Guests.

Code Example 44: SaveResultSet transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <SaveResultSetRequest>
      <ResultSetID>R1002_RS_978991090832</ResultSetID>
      <newResultSetID>result1</newResultSetID>
   </SaveResultSetRequest>
</CatalogService>
```

3.7.13 ListServicesForGranule

This transaction lists the name and description of services available for the specified granule.

3.7.14 PresentSearchOptions

This transaction requests a list of the settable options available for the Catalog Service. A user may request particular options by name; otherwise, all settable option definitions will be returned. The transaction is not currently implemented.

3.7.15 PresentSearchPreferences

This transaction requests a list of the user-selected and/or ECHO default preference settings for the Catalog Service. The transaction is not currently implemented.

3.7.16 UpdateSearchPreferences

This transaction updates the user's default search preferences. These preferences may be overridden in any given Query Request; however, setting these properties removes the necessity of providing this information as part of every Query Request. These preference values will be stored across sessions for registered users, but will expire at the end of a session for guest users. The transaction is not currently implemented.

3.8 Error Messages

Table 3-6 gives the possible error messages associated with each transaction within the Catalog Service. Some error messages are more complex then others and so will be treated in more detail in the following subsections.

Table 3-6: Catalog service error messages.

Transaction	Error Message	Description
IIMSAQL query specified is not well formed or does not conform to its DTD	See Section 3.3.2.1.1 below.	See Section 3.3.2.1.1 below.
Provider specified under the dataCenterId element does not exist	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 10,column 26 Invalid data center ID [ORNL2] . DataCenter Id must be a valid ID NOT enclosed in quotes. Valid Ids are [ORNL,TEST_PROVIDER2,TEST_PROVIDER1])	The error message lists all the valid providers that can be specified as dataCenterIds. This list will vary according to the providers registered within ECHO at the time the query was issued.
Strings and text patterns specified for searches on campaign, dataSetId, spatialKeypwords etc., not enclosed in single quotes	See Section 3.3.2.1.2 below.	See Section 3.3.2.1.2 below.
Invalid date specified for any search element that requires dates like temporal, echoLastUpdate etc.	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 18,column 47 Invalid Date)	Check whether the date is valid like for example date is trying to search on 29th of February.
Coordinates specified for spatial window points not within the spatial limits of the earth	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 17,column 44 latitude value of IIMSPoint must be a number between [-90,90])	The error message will change whether the spatial window is being specified using Polygon, MultiPolygon etc., and also whether the latitude or longitude is out of range.
Cloud cover range specified not within 0-100	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 16,column 38 Invalid range for element cloudCover. Attribute 'upper' of range element must be a number between [0,100])	Indicates that the Upper range specified is beyond 100.
Values specified for any ranges not conforming to upper>= lower	gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 16,column 38 Invalid range for element cloudCover.)	
Information related to presenting the results (when the request message specifies ResultType as RESULTS)	See Section 3.3.2.1.4 below	See Section 3.3.2.1.4 below.

3.8.1 Query Error Messages

3.8.1.1 IIMSAQL query specified is not well formed or does not conform to its DTD

If the request message sent to ECHO for any transaction in the CatalogService is not well formed XML or does not conform to its DTD an attempt will be made to return the XML parser generated error message.

Error Message 3: Parser generated error when the "for" element is not specified in an IIMSAQL query.

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in at line 16, column 9:The
content of element type "query" must match "(for,dataCenterId,where)".)
```

Error Message 4: Parser generated error when the IIMSAQL query fails to have correct matching tags

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed
Error in  at line 14, column 41:The element type "value" must be terminated by the matching end-tag "</value>".)
```

But when the system attempts to return this error message, the error message itself invalidates the XML that is being returned to the user (because of the ‘<’ and the ‘>’ characters that are not escaped) and the ECHO system internally throws another error message. So the error message that the user will see will be the following.

Error Message 5: XML message generated by internal ECHO error.

```
<?xml version="1.0" standalone="no" ?>
<SessionManager>
   <Error>
      <![CDATA[
         Stopping after fatal error: The element type "Message" must be terminated by the matching end-tag
         "</Message>".
      ]]>
   </Error>
</SessionManager>
```

This is not the best error message to be sending to the user and future versions will correct this problem. In order to minimize the possibility of receiving such vague error messages, make sure the query in IIMSAQL is validated independently with its own DTD before embedding it in the CDATA section of the QueryRequest message.

3.8.1.2 Strings and text patterns specified for searches on campaign, dataSetId, spatialKeypwords etc., not enclosed in single quotes

There are two possible error messages – one for string values and one for text patterns.

Error Message 5: String value error message for searches not enclosed in single quotes.

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 14,column 37
campaign values must be strings enclosed in single-quotes.)  or
```

Error Message 7: Text pattern error message for searches not enclosed in single quotes.

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Query validation Failed Error in query at line 14,column 49
textPattern element of campaign must be a string enclosed in single-quotes.)
```

The word “campaign” in the error message will be replaced by the search criteria that is violating this rule

3.8.1.3 Spatial window specified as part of the spatial query is invalid

If a spatial window specified in the spatial query is invalid then an error message will be returned.

Error Message 8: Invalid spatial window error.

```
gov.nasa.echo.business.query.IIMSAQL.IIMSAQLQueryException(Invalid Window::Oracle error:13367)
```

This may not seem a very useful error message because it does not clearly state what about the spatial window is invalid. The Oracle error code 13367 is an indication of the problem, but the client would not be able to link this code to a useful error message. We will attempt to list the error codes and the corresponding explanation. This list may not be complete. If you receive an error code that is not listed, please let us know.

Table 3-7: Spatial error codes.

Oracle Spatial Error Code	Explanation of the error code
13349	The boundary of a polygon intersects itself. Correct the geometric definition of the spatial window
13351	Two or more rings of a complex polygon overlap. The inner or outer rings of a complex polygon overlap. All rings of a complex polygon must be disjoint. Correct the geometric description of the object.
13356	There are repeated points in the sequence of coordinates. Remove the redundant points.
13367	The orientation of the rings in the spatial window is incorrect. Make sure all the rings are in counter-clockwise order.

3.8.1.4 Information related to presenting the results (when the request message specifies ResultType as RESULTS)

The error messages in this case would be the same as those during Present transaction that are related to TupleTypes, IteratorSize and Cursor values. Please refer to section 3.6.3.

3.8.2 Query and Results Management Error Messages

3.8.2.1 PresentResults

Error Message 9: PresentResults error returned when user tries to present a Result Set that has not been created in the current session or previously saved.

```
gov.nasa.echo.services.ejb.catalogservice.UserResultNotFoundException(result=[ResultX]
for userId=[UserY] not found)
```

Where ResultX is the ResultSetID that was requested to be presented and UserY is the userId of the registered user or a temporary Id assigned to the Guest.

Error Message 10: PresentResults error returned when invalid TupleTypes are used.

```
Attributes [ShortN, DataSetID, ECHOUpdateDate] not valid for QueryType=[COLLECTIONS] and DTDType=[ECHO]
```

The DTDType by default is ECHO. The valid TupleTypes for the Present depend on the DTDType selected (current options are ECHO and BMGT) and whether the results are for granules or collections. If the user specifies a TupleType that is not valid for this particular combination chosen the error message returned will include the list of invalid TupleTypes.

For example, the error message shown above indicates the situation when trying to present results for Collections using the ECHO DTDType. We mistyped ShortName as ShortN, write DataSetID instead of DataSetId and write ECHOLastUpdate as ECHOUpdateDate.

3.8.2.2 Invalid values for either the Cursor or IteratorSize

If invalid values are specified for Cursor and Iterator then no error is generated, instead ECHO makes certain assumptions for the user:

1. In case the Cursor or Iterator values are negative numbers then the default results are returned (i.e. Cursor starting from 1 and Iterator size of 10)
2. In case the Cursor value is beyond the result set size, payload will contain the header of the results in XML but it will not actually have any results in it.

Error Message 11: Error returned when invalid values for either the Cursor or IteratorSize are used

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "http://api.echo.eos.nasa.gov/echo/dtd/ECHOCollectionResults.dtd">
<results/>
```

In case the Iterator size takes the cursor beyond the total number of results, results are shown up to the last result in the result set.

3.8.2.3 SaveQuery

Error Message 12: SaveQuery - Error returned when a Guest is trying to save a query

```
Transaction available only to Registered Users
```

3.8.2.4 SaveResult

Error Message 13: SaveResult - Error returned when a Guest is trying to save a query

```
Transaction available only to Registered Users
```

Error Message 14: SaveResult - Error returned when a user tries to save a result with an existing name

```
Result with [savedName] name already exists
```

Error Message 15: SaveResult - Error returned when a user tries to save a result that does not exist yet

```
Result with name [resultSetId] does not exist for user [User101]
```

3.8.2.5 PresentSavedQuery

Error Message 16: PresentSavedQuery - Error returned when a Guest tries to present a saved query

```
Transaction available only to Registered Users
```

Error Message 17: PresentSavedQuery - Error returned when a user tries to present a saved query that does not exist yet

```
Query with [queryName] name does not exist
```

3.8.2.6 RemoveSavedQuery

Error Message 18. RemoveSavedQuery - Error returned when a Guest tries to remove a saved query

```
Transaction available only to Registered Users
```

Error Message 19: RemoveSavedQuery - Error returned when a user tries to remove a saved query that does not exist yet

```
Query with [queryName] name does not exist
```

3.8.2.7 Remove Saved ResultSet

Error Message 20: Remove Saved ResultSet - Error returned when a user tries to remove a result set that does not exist

```
Saved Result [myresult] for userId=User101 not found
```

3.8.2.8 List Saved Queries

Error Message 21: List Saved Queries - Error returned when a Guest tries to list saved queries

```
Transaction available only to Registered Users
```

3.8.2.9 List Saved ResultSets

Error Message 22: List Saved ResultSets - Error returned when a Guest tries to list saved result sets

```
Transaction available only to Registered Users
```

3.9 Alternative Query Language

ECHO Alternative Query Language (IIMSAQL) is a query language that provides for searches on collections (Discovery) and searches on granules (Data or Inventory search) on the ECHO site. IIMSAQL is an XML based language. The DTD for IIMSAQL can be found at IIMSAQLQueryLanguage.dtd.

Code Example 45: General AQL structure.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
   <for value= "collections" or "granules" />
   <dataCenterId>
      <all/>
   </dataCenterId>
   <where>
      <list of conditions ... >
   </where>
</query>
```

Note: The <for/> tag shown in this example does not require a value attribute to be set, though it should be kept in mind that if you do not explicitly set this attribute it will default to "collections." If you do not set the value attribute your query will validate correctly, but an error can still be experienced if you later use granule settings for other attributes in your query.

The query language is designed to be data driven and declarative. The DTD defines the fields that can be queried as well as the specific data centers that should be used for the query. Under the <dataCenterId> tag, the user defines the data centers from which to search. Then for each field that can be queried, the user can specify the value (or set of values) that must (or one of which must) be satisfied by each returned result. The fields that can be searched on for collections occur as children of <collectionCondition> element. The fields that can be searched on for granules occur as children of <granuleCondition> element. Only those granules/collections that satisfy all the conditions form part of the result (i.e., ANDing of the conditions in the query is implied.). The list of conditions consists of <collectionCondition> (or <granuleCondition>) elements for Discovery (or Inventory search).

The DTD for IIMSAQL specifies that the <where> tag can have any number of <collectionCondition> or <granuleCondition> elements. Moreover, it further specifies that each of these tags can contain any one of the search elements described below. However, note that for the query to be valid each of the search elements may appear ONLY once.

Code Example 46: Discovery search.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query><for value="collections"/>
   <dataCenterId><value>ORNL-DAAC</value></dataCenterId>
   <where>
      <collectionCondition> … </collectionCondition>
      <collectionCondition> … </collectionCondition>
      …
   </where>
</query>
```

Code Example 47: Inventory search.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query><for value="granules"/>
   <dataCenterId><value>ORNL-DAAC</value></dataCenterId>
   <where>
      <granuleCondition> … </granuleCondition>
      <granuleCondition> … </granuleCondition>
      …
   </where>
</query>
```

The data centers can be specified by explicitly listing them or by using the empty element <all/> to indicate that all the data centers should be searched. This is the default condition. i.e., if no condition is specified for <dataCenterId> element all data centers are searched. Do not enclose data center IDs in single-quotes.

Code Example 48: dataCenterID search element. Look for collections/granules in the ORNL provider at ECHO only.

```
<dataCenterId><all/></dataCenterId>

Search all data providers.

<dataCenterId><list>
   <value>ORNL-DAAC</value>
   <value>LPDAAC_ECS</value>
</list></dataCenterId>

Look for Collections/Granules in the LPDAAC_ECS or ORNL-DAAC data at ECHO only.

<dataCenterId>
    <value>ORNL-DAAC</value>
</dataCenterId>
```

3.9.1 Discovery Search

Discovery is performed using the search for Collections. Searches for collections must enclose the search criteria in <collectionCondition> element. This element takes an attribute negated that should be set to 'Y' if you want the search to include the negated criteria. E.g. If you want to search for all collections except the ones with processing level 1 or 2 you can form the search like that shown in the accompanying example.

Code Example 49: CollectionCondition element.

```
<collectionCondition negated='Y'>
  <processingLevel><list>
     <value>'1'</value>
     <value>'1A'</value>
     <value>'1B'</value>
     <value>'2'</value>
  </list></processingLevel>
</collectionCondition>
```

The negated attribute can be set to 'Y' for any criterion unless specified otherwise. The following criteria can be used to search for Collections.

Table 3-8: Discovery search criteria.

Search Criteria	XML Element
CampaignShortName	`<CampaignShortName>...</CampaignShortName>`
Dataset ID	`<dataSetId>...</dataSetId>`
ECHO Insert Date	`<ECHOInsertDate>...</ECHOInsertDate>`
ECHO Last Update	`<ECHOLastUpdate>...</ECHOLastUpdate>`
Online Collections Only	`<onlineOnly/>`
Parameter	`<parameter>...</parameter>`
Processing Level	`<processingLevel>...</processingLevel>`

Sensor Name	`<sensorName>...</sensorName>`
Short Name	`<shortName>...</shortName>`
Source Name	`<sourceName>...</sourceName>`
Spatial	`<spatial>...</spatial>`
Spatial Keywords	`<spatialKeywords>...</spatialKeywords>`
Temporal	`<temporal>...</temporal>`
Temporal Keywords	`<temporalKeywords>...</temporalKeywords>`
PSA Names	`<psaNames>...</psaNames>`

3.9.2 *Inventory Search*

Inventory search is performed for searching Granules. Searches for granules must enclose the search criteria in <granuleCondition> element. As in the case of collectionCondition, this element takes an attribute negated that should be set to 'Y' if you want the search to include the negated criteria. The negated attribute can be set to 'Y' for any criterion unless specified otherwise. Granules can be searched using the following criteria that are further discussed in the following sections.

Table 3-9: Inventory search criteria.

Search Criteria	XML Element
Only Granules with Browse Data	`<browseOnly/>`
CampaignShortName	`<CampaignShortName>...</CampaignShortName>`
Percentage of Cloud Cover	`<cloudCover>...</cloudCover>`
Dataset ID	`<dataSetId>...</dataSetId>`
ECHO Insert Date	`<ECHOInsertDate>...</ECHOInsertDate>`
ECHO Last Update	`<ECHOLastUpdate>...</ECHOLastUpdate>`
Either Day or Night Granules Only	`<dayNightFlag/>`
Only Global Granules	`<globalGranulesOnly/>`
Search on echo granule Ids (formerly granuleId)	`<ECHOGranuleID>...</ECHOGranuleID>`
Search on Granule UR (provider specific)	`<GranuleUR>...</GranuleUR>`
Online Granules Only	`<onlineOnly/>`
Path Row Range	`<pathRow>...</pathRow>`
LocalGranuleID	`<LocalGranuleID>...</LocalGranuleID>`
PSA	`<psa>...</psa>`
Sensor Name	`<sensorName>...<sensorName>`
Source Name	`<sourceName>...</sourceName>`
Spatial	`<spatial>...</spatial>`
Temporal	`<temporal>...</temporal>`

3.9.3 Common Search Elements

3.9.3.1 CampaignShortName

Name(s) of the campaign/project that gathered data associated with the collection/granule. CampaignShortName can take a single-quoted string in a <value>, a list of single-quoted strings in a <list> or a textPattern in a <textPattern> element as defined in section on Miscellaneous Elements.

Refer to the sub-section on textPattern under Miscellaneous Elements for detailed explanation on the pattern string. Later versions may support text patterns from Oracle InterMedia.

Code Example 50: CampaignShortName search element.

```
<CampaignShortName>
   <value>'River'</value>
</CampaignShortName>

That is transformed to campaignShortName = 'River'

<CampaignShortName>
   <list>
      <value>'River'</value>
      <value>'Lake'</value>
   </list>
</CampaignShortName>

That is transformed to campaignShortName IN ('River', 'Lake')

<CampaignShortName>
   <textPattern operator="LIKE">'AS_A%D'</textPattern>
</CampaignShortName>

That is transformed to campaignShortName LIKE 'AS_A%D'.
```

3.9.3.2 DatasetID

This specifies the universal name of a collection. This element can be used to restrict the search to a few Collections. It can be specified using the <value> element to specify a single Collection, a <list> for a list of Collections or a <textPattern>. You can use the <value> or <list> only if you know the name of the Collection. This can be the case when you have performed a Discovery search and are now performing an Inventory search for granules within the collections of your interest.

Note: Beginning with ECHO release 5.5, the use of DatasetID is case sensitive. If the exact DatasetID is not used, searches could return incorrect results (including 0 hits).

Code Example 51: DatasetID search element. Look for Granules in the data sets 'LEAF CHEMISTRY, 1992-1993 (ACCP)' or 'ASTER DEM Product V002'only.

```
<collectionCondition>
   <dataSetId>
     <textPattern>'%1A'</textPattern>
   </dataSetId>
<collectionCondition>

Search for all data sets that end with '1A'
```

```
<granuleCondition>
   <dataSetId><list>
     <value>'LEAF CHEMISTRY, 1992-1993 (ACCP)'</value>
     <value>'ASTER DEM Product V002'</value>
   </list></dataSetId>
</granuleCondition>
```

3.9.3.3 OnlineOnly

Used to specify search for granules or collections that are available online. Negated condition is not supported.

Code Example 52: OnlineOnly element.

```
<collectionCondition>
   <onlineOnly/>
<collectionCondition>
```

3.9.3.4 ECHO Insert Date

Used to specify when a collection or granule was inserted into ECHO. This can be specified only as date range. Negated condition is not supported.

Code Example 53: ECHOInsertDate element.

```
   <ECHOInsertDate>
       <dateRange>
         <startDate>
         <Date YYYY="2001" MM="04" DD="05"/>
         </startDate>
       </dateRange>
   </ECHOInsertDate>
```

3.9.3.5 ECHO Last Update

Used to specify when a collection or granule was last updated inside ECHO. This can be specified only as date range. Negated condition is not supported.

Code Example 54: ECHOLastUpdate element.

```
   <ECHOLastUpdate>
      <dateRange>
         <startDate>
           <Date YYYY="2001" MM="01" DD="01"/>
         </startDate>
         <stopDate>
           <Date YYYY="2001" MM="06" DD="01"/>
         </stopDate>
      </dateRange>
   </ECHOLastUpdate>
```

3.9.3.6 Sensor Name

Names of sensors associated with a collection or granule. The sensor names can be specified as one single-quoted value, a list of single-quoted values or a textPattern. If a list or a single value is specified then the search looks for an exact match between the string specified and the data stored. Hence, if an exact match is not found, no data is returned. In that case it is more useful to use <textPattern> element for the search. If you know the exact names of the sensors from previous searches or list of valids then the single value or list of values is a more efficient way to search for the sensors.

Code Example 55: SensorName element.

```
<sensorName><value>'etm+'</value></sensorName>
```

3.9.3.7 Source Name

Names of platforms associated with a collection or granule. The same rules as for sensor names apply here.

Code Example 56: SourceName element.

```
<sourceName><value>'L7'</value></sourceName>
```

3.9.3.8 Spatial

Spatial is used to restrict the search within a spatial extent of the earth. Negated condition is not supported. This condition does not allow the negated attribute of the granuleCondition or collectionCondition element to be 'y'. i.e., queries of the kind: all granule/collection NOT in this spatial region, are not allowed. Specifying this condition and globalGranulesOnly criterion in the same query is not allowed.

The spatial extent can be specified using the OpenGIS Consortium's (OGC) Geography Markup Language (GML) elements <Polygon>, <MultiPolygon> or <LString> found in GeometryCollection.dtd or by using the ECHO elements <IIMSPolygon>, <IIMSMultiPolygon> or <IIMSLine>. While using GML elements, the point MUST be specified with longitude first followed by the latitude.

A Polygon consists of one or more rings. Some rings can be embedded within others that define the interior of the polygon. The ECHO system can handle polygons with only one external ring and multiple internal rings.

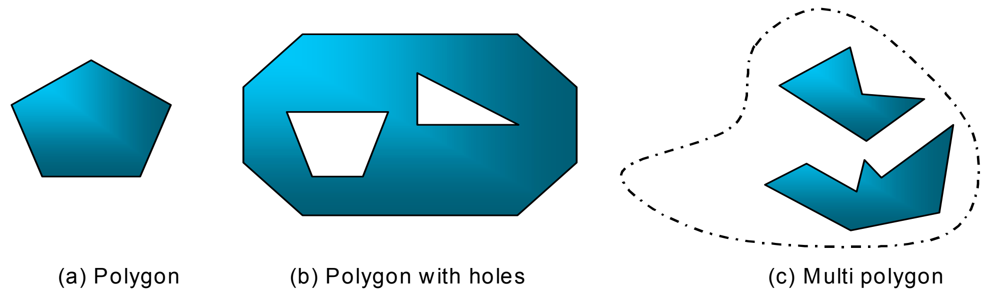


Figure 3-1 Allowed geometries for spatial data and query windows.

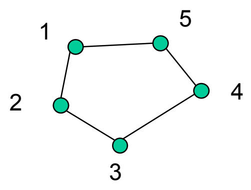


Figure 3-2. The points of a polygonal ring should be specified in counter clockwise order.

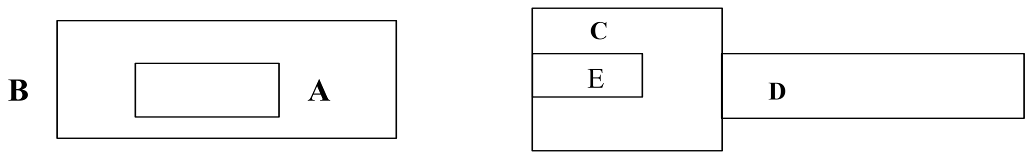
In order to define a ring the last point in the ring should be the same as the first point in the ring. It is necessary to specify the points in each polygonal ring in counter-clock wise order.

In order to specify multiple outer rings the <MultiPolygon> or <IIMSMultiPolygon> elements should be used, with each outer ring in a separate <Polygon> (<MultiPolygon>) or <IIMSMultiPolygon> element (<IIMSMultiPolygon>). There should be no overlap between any of the outer rings defined.

<IIMSLine> or <LString> are used to specify a line on the earth. It is necessary to specify the points in the line from one end point to the other.

Spatial Operators available in the ECHO System are described below. Relationship between objects A and B is defined to be:

- **EQUAL.** They have the same boundary and interior.
- **TOUCH.** The boundaries intersect but the interiors do not.
- **WITHIN.** Interior and boundary of the first object is completely contained in the interior of the second.
- **CONTAINS.** The interior and boundary of the second object is completely contained in the interior of the first.
- **RELATE.** The objects are non-disjoint, i.e., their body and/or boundaries intersect.



(a) Search region B, A is within; Search region A, area B contains A

(b) C and D touch; C and E do not touch

Figure 3-3. Types of spatial relationships.

The query is defined such that the user specified spatial extent, is the second object in the query. Hence use CONTAINS if you want to retrieve granules/collections that contain the specified polygon, use WITHIN if you want to retrieve granules/collections that are within the specified spatial extent and use RELATE to retrieve granules/collections that overlap in some way. RELATE is the default operator.

In the latest ECHO system, ECHO supports multiple spatial representations (or spatial types). Spatial data of different spatial representation are described differently, and hence are stored and queried in different way. The spatial types that ECHO currently supports are described below.

3.9.3.8.1 Normal

Conventional spatial data are described as a certain shape such as a polygon, a multi-polygon, or a line. The spatial type for this type of data is categorized as "NORMAL". Spatial data of this type are stored in the database as an Oracle object to record its shape, spatial locations and coordinate system used, Cartesian or Geodetic.

Because an Oracle search for spatial data has different restrictions for different coordinate system, some issues need to be noted. For the Cartesian system, Oracle allows the coverage to be as large as the whole earth but does not allow data that covers the poles or crosses the date line. To search this type of data, the search window needs to conform to these restrictions. For the Geodetic system, Oracle allows the data to cross the date line and the poles but the total area of a single polygon cannot exceed half the Earth. Again, the search window for such data needs to conform to these constraints. Oracle does not validate the spatial search window before it executes the query. ***If the window happens to be invalid no error is generated but the results produced may be incorrect.*** In order to reduce the effect of this issue on clients, ECHO validates the spatial window. Since the window validation is performed whether the data searched is Geodetic or Cartesian, the validation routine only validates the window to conform to the Geodetic coordinate system. ***It is the client's responsibility to make sure that they conform to the extra***

restriction imposed when querying the Cartesian system data. In addition, even though the Cartesian system allows the query window to be as large as the whole earth, ECHO will restrict the window to be no larger than half the area of the earth. Smaller sized spatial search windows result is faster response times.

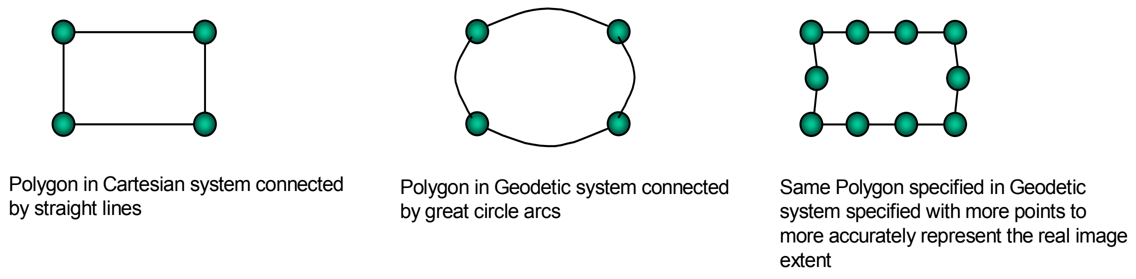

Polygon in Cartesian system connected by straight lines
Polygon in Geodetic system connected by great circle arcs
Same Polygon specified in Geodetic system specified with more points to more accurately represent the real image extent

Figure 3-4. Polygon area defined depends on the number of points and the coordinate system used.

The actual area on the earth that is queried depends on the coordinate system used by the provider and the number of points in the query window. When searching data from providers using Cartesian coordinates the search polygon is converted to a Cartesian polygon with points connected using straight lines, The search polygons is converted to a Geodetic polygon while searching data from providers using Geodetic coordinate system and the points are connected using great circle arcs. This may result in slightly different results for data from providers using Cartesian and those using Geodetic coordinate system. To ensure better accuracy, represent the search window with more than just the corner points. Experiments have shown acceptable performance with 50-point polygons.

Oracle Spatial search is a two-step process. The first step filters the data by querying the spatial window against the bounding box of the spatial data. The second step is a slower more accurate polygonal geometry search for exact results on the results from the first step. A small window results in more data being eliminated by the first filter, so the search will run faster.

Skewed polygonal windows or data will have lots of wasted space and the bounding box will result in more false positives passing through the fast filter (see Figure 3-5).

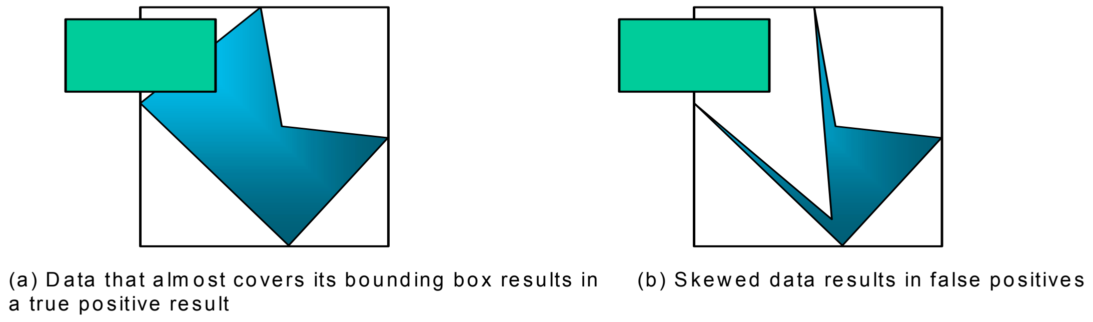

(a) Data that almost covers its bounding box results in a true positive result
(b) Skewed data results in false positives

Figure 3-5. Data orientation with respect to its bounding rectangle may play a role in the response time of a spatial query.

3.9.3.8.2 Global

A special case is that the spatial coverage of the data is the entire earth. The data does not come with any specific spatial value but rather a flag to denote that its spatial extent is global. Currently ECHO only supports search on granules.

3.9.3.8.3 Orbit

Spatial data can be orbit-based and is generated from satellites running along orbits. Because an orbit is a fixed direction with a fixed declination angle and the covered area is a stripe of fixed width, the spatial coverage of a granule can be derived from the granule's startng and ending latitude, and the equator-crossing longitude where its orbit originally starts. Thus, orbit-based spatial data is not stored as a shape object but instead as the original data. In searching for orbit-based data the spatial coverage can be as large as the entire earth surface including polar areas. ECHO performs a simple validation against the spatial window to make sure it does not exceed the extent of the earth (–90 to 90 degrees latitude and -180 to 180 degress longitude).

Note: ECHO only supports searching on single-orbit granules and the only supported spatial operator is RELATE.

The <SpatialType> tag is used to specify the spatial type. If the user opts not to specify this field, ECHO performs a default search on NORMAL (Cartesian and Geodetic), GLOBAL and ORBIT.

Code Example 57: Spatial Search for Orbit-based data per user specified spatial window.

```
<spatial operator="RELATE">
  <Polygon>
    <LRing>
      <CList>-120, -30, -100, -60, 5, -90,
      -120, -60, 160, 5, 160, 60, 120, 85,
      5, 85, -120, 30, -120, -30</CList>
    </LRing>
    <LRing>
      <CList>80, 20, 80, 60, 20, 60,
             20, 20, 80, 20</CList>
    </LRing>
  </Polygon>
  <SpatialType>
     <list>
      <value>ORBIT</value>
    </list>
  </SpatialType>
</spatial>
```

Code Example 58: A spatial query with a Polygon with an external and an internal Ring defined using ECHO elements.

```
<spatial operator='RELATE'>
   <IIMSMultiPolygon>
      <IIMSPolygon>
         <IIMSLRing>
            <IIMSPoint lat="85" long="-50"/>
            <IIMSPoint lat="70" long="-60"/>
            <IIMSPoint lat="60" long="-50"/>
            <IIMSPoint lat="70" long="-40"/>
            <IIMSPoint lat="85" long="-50"/>
         </IIMSLRing>
      </IIMSPolygon>
      <IIMSPolygon>
         <IIMSLRing>
            <IIMSPoint lat="-85" long="-50"/>
            <IIMSPoint lat="-70" long="-40"/>
            <IIMSPoint lat="-60" long="-50"/>
            <IIMSPoint lat="-70" long="-60"/>
            <IIMSPoint lat="-85" long="-50"/>
         </IIMSLRing>
      </IIMSPolygon>
   </IIMSMultiPolygon>
</spatial>
```

Code Example 59: A MultiPolygon spatial query defined using GML elements.

```
<spatial operator='EQUAL'>
  <LString>
    <CList>70, 80, 80, 80, 90, 90</CList>
  </LString>
</spatial>
```

3.9.3.9 Temporal

Used to specify the temporal extent of the data. The dates are stored as Gregorian dates and times are stored in GMT. This search allows only dates in YYYY-DD-MM format. A range of time as beginning and ending dates can be specified or a periodic temporal range can be specified as start date, end date and the start and end day of each year can be specified. The start and end Day should be an integer between 1 and 365 with start day <= end day. End day can be specified as 366 but if the temporal range contains a non-leap year then an error will be generated.

Code Example 60: Temporal search element.

```
<temporal>
   <startDate><Date YYYY="1990" MM="5" DD="12"/></startDate>
   <stopDate><Date YYYY="1992" MM="6" DD="13"/></stopDate>
</temporal>

Searches for granule/collections whose temporal range overlaps with the time between May 12, 1990 to June 13, 1992.

<temporal>
   <startDate><Date YYYY="1990" MM="5" DD="12"/></startDate>
   <stopDate><Date YYYY="1992" MM="2" DD="10"/></stopDate>
   <startDay value="5"/>
   <endDay value="60"/>
</temporal>
```

Searches for granules/collections whose temporal range overlaps with the time ranges: May 16, 1990 - July 14, 1990 or January 5, 1991 - March 5, 1991 or January 5, 1991 - February 10, 1991.

3.9.4 Search Elements Specific to Discovery

3.9.4.1 Parameter

Specify the Geophysical terms associated with a collection. This can be specified only for Discovery not for Inventory search. Later version may provide for parameter search at the granule level. If a list or a single value is specified then the search looks for an exact match between the string specified and the data stored. Hence if the search is on 'Imagery' and the data stored is 'Satellite Imagery' the data is not returned since there was not exact match. Hence it is more useful to use <textPattern> element for the search.

Code Example 61: Parameter search element.

```
<parameter>
   <textPattern operator="LIKE">'%Imagery%'</textPattern>
```

```
</parameter>
```

3.9.4.2 Processing Level

Level to which the Data (collection/granule) has been processed. Possible values are 0, 1, 1A, 1B, 2, 3, 4. The Search can specify one or more processing Levels. The values must be single-quoted strings.

Code Example 62: Processing Level search element. Search for all collections at processing level 1 or 1A or 1B.

```
<collectionCondition>
   <processingLevel>
      <list>
         <value>'1'</value>
         <value>'1A'</value>
         <value>'1B'</value>
      </list>
   </processingLevel>
</collectionCondition>
```

Code Example 63: Processing Level search element. Search for all collections at processing level 1 or 1A or 1B and, 1C etc if they exist. This also returns results for collections with processing levels 13, 14, etc.

```
<collectionCondition>
   <processingLevel>
      <textPattern
        operator="LIKE">'1_'</textPattern>
   </processingLevel>
</collectionCondition>
```

3.9.4.3 PSA Names

Search for collections that contain certain Provider Specific Attributes. The possible PSAs will change from one data center to another. The search can specify one or more PSAs. The values must be single-quoted strings.

Code Example 64: PSA Names search element.

```
<collectionCondition>
  <psaNames>
    <list>
      <value>'PSA_1'</value>
      <value>'PSA_3'</value>
    </list>
  </psaNames>
</collectionCondition>
```

3.9.4.4 Short Name

Short name of a collection. May or may not be unique for a particular provider.

Code Example 65: Short Name search element. Search for all collections that have shortNames starting with 'BOREAS.'

```
<collectionCondition>
   <shortName>
      <textPattern>'BOREAS%'</textPattern>
   </shortName>
</collectionCondition>
```

3.9.4.5 Spatial Keywords

Specifies a word or phrase that summarizes the spatial region covered by the collection. A collection may cover several regions.

Code Example 66: Spatial Keywords search element – Search for all collections that cover Africa or Bermuda or the Indian Ocean..

```
<collectionCondition>
   <spatialKeywords>
      <list>
         <value>'Africa'</value>
         <value>'Bermuda'</value>
         <value>'Indian Ocean'</value>
      </list>
   </spatialKeywords>
</collectionCondition>
```

Code Example 67: Spatial keywords search element – Search for all collections that cover some regions of the Americas.

```
<collectionCondition>
   <spatialKeywords>
     <textPattern  operator="LIKE">'%america%'</textPattern>
   </spatialKeywords>
</collectionCondition>
```

3.9.4.6 Temporal Keywords

Specifies a word or phrase that summarizes the temporal characteristics referenced in the collection.

Code Example 68: Temporal keywords search element – Search for all collections that have data during the spring or fall.

```
<collectionCondition>
   <temporalKeywords>
      <list>
         <value>'spring'</value>
         <value>'fall'</value>
      </list>
   </temporalKeywords>
</collectionCondition>
```

3.9.5 Search Elements Specific to Inventory

3.9.5.1 Only Granules with Browse Data

Used to specify that only granules that have browse data available should be returned. Negated condition is not supported. i.e. If the parent granuleCondition element has the attribute negated='y' the NOT operator is not added and the condition remains positive.

Code Example 69: Only Granules with browse data search element.

```
<granuleCondition><browseOnly/>
</granuleCondition>
```

```
or

<granuleCondition><browseOnly value="Y"/>
</granuleCondition>
```

3.9.5.2 Percentage of Cloud Cover

Range of percentage of scene cloud coverage in a granule. The value in each attribute of range should be a float between [0,100] that specifies the range in percentage of cloud coverage in a granule that is returned. Negated condition is not supported.

Code Example 70: Percentage of Cloud Cover search element. Return only those granules that have cloud cover between 10% and 20%.

```
<granuleCondition>
   <cloudCover>
     <range lower="10" upper="20"/>
   </cloudCover>
</granuleCondition>
```

3.9.5.3 Either Day or Night Granules Only

Specifies that granules that are gathered during daylight or nighttime or both, be returned. If this criterion is not specified then granules gathered at any/all time are returned. Negated condition is not supported. i.e. If the parent granuleCondition element has the attribute negated = 'y' the NOT operator is not added and the condition remains positive.

Code Example 71: Either Day or Night Granules Only search element. Select granules gathered at least partly during daylight.

```
<granuleCondition>
   <dayNightFlag value="DAY"/>
</granuleCondition>
```

Code Example 72: Either Day or Night Granules Only search element. Select granules gathered at least partly during the night.

```
<granuleCondition>
   <dayNightFlag value="NIGHT"/>
</granuleCondition>
```

Code Example 73: Either Day or Night Granules Only search element. Select granules gathered partly in night and partly during the daylight.

```
<granuleCondition>
   <dayNightFlag value="Both"/>
</granuleCondition>
```

3.9.5.4 Only Global Granules

Search only for Global granules. i.e., granules that span the whole earth. Do not specify both, the spatial criteria and globalGranulesOnly. Only one of the criteria is allowed. Negated condition is not supported. This condition does not allow the negated attribute of the granuleCondition element to be 'y'. i.e. queries of the kind: "all granules NOT global", are not allowed.

Code Example 74: Only Global Granules search element.

```
<granuleCondition>
   <globalGranulesOnly/>
</granuleCondition>

or

<granuleCondition>
   <globalGranulesOnly value="Y"/>
</granuleCondition>
```

3.9.5.5 Search on ECHO Granule IDs

Specify an inventory search for specific granules. This search does not need to specify any spatial or temporal constraint. This can be used as a second stage query when the user knows the list of granules he is interested in from a previous query. The search is then limited to granules in the list. The list may contain granules from different data sets. The search is performed on ECHO_GRANULE_ID; an ECHO wide unique ID. Negated condition is not supported. i.e. If the parent granuleCondition element has the attribute negated ='y' the NOT operator is not added and the condition remains positive. The names of the granules must be enclosed in single quotes.

Code Example 75: Search on ECHO Granule IDs search element.

```
<granuleCondition>
   <ECHOGranuleID>
      <list>
         <value>'G1984-ORNL'</value>
      </list>
   </ECHOGranuleID>
</granuleCondition>
```

3.9.5.6 Search on Granule UR

Specify an inventory search for specific granules. Like granule IDs, this criterion can be used as a secondary stage query. Instead of using the ECHO specific id for the granules, this criterion allows the user to use the provider given name for the granules. These names are more meaningful and easier to remember. Since the granule names given by the provider are not guaranteed to be unique within ECHO, if a granule with the same name exists for more than one dataCenterId specified in the query, all will be returned in the results. Negated condition is not supported. i.e. If the parent granuleCondition element has the attribute negated ='y' the NOT operator is not added and the condition remains positive. The names of the granules must be enclosed in single quotes.

Code Example 76: Search on Granule UR search element.

```
<granuleCondition>
   <GranuleUR>
      <list>
         <value>'ACCP_CANOPYCHEM.df_can_calc.dat'</value>
      <value>'ANGLIA_10YRCLIMATE.cfrs1120.zip'</value>
    </list>
  </GranuleUR>
</granuleCondition>
```

3.9.5.7 Path Row Range

This is an alternative to specifying the temporal range. A negated condition is not supported. (i.e., if the parent element has the attribute negated='y' the NOT operator is not added and the condition remains positive).

The Path Row criterion needs three elements: the WRS Path Type whose values can be 1 or 2, the path range and the row range. The path and row may also take a single value. It returns all granules that are specified with the user specified WRS Type and overlap with the path and row range specified. Path range and Row range must have both lower and upper attributes specified.

Code Example 77: Path Row Range search element.

```
<granuleCondition>
   <pathRow>
      <path><range lower="15" upper="15"/></path>
      <row><range lower="33" upper="33"/></row>
      <wrsType value="1"/>
   </pathRow>
 </granuleCondition>
```

3.9.5.8 LocalGranuleID

Specify an inventory search for specific granules. This represents the ID assigned to this data by the Science team that produced this data. Instead of using the ECHO specific IDs for the granules, this criterion allows the user to use the producers given name for the granules. These names are available to the Science teams and are easier for them to search on. Since the granule names given by the producer are not guaranteed to be unique within ECHO, if a granule with the same name exists for more than one dataCenterId specified in the query, all will be returned in the results. Negated condition is not supported. i.e. If the parent granuleCondition element has the attribute negated ='y' the NOT operator is not added and the condition remains positive. The names of the granules must be enclosed in single quotes.

Code Example 78: LocalGranuleID search element.

```
<granuleCondition>
  <LocalGranuleID>
    <list>
      <value>'MOD03.A2000107.0930.003.2002056052717.hdf'</value>
      <value>'MOD01.A2000107.0740.003.2002056051321.hdf'</value>
    </list>
  </LocalGranuleID>
</granuleCondition>
```

3.9.5.9 PSA

Specify an inventory search for specific granules that contain certain Provider Specific Attributes and the exact value that the PSA may hold. The actual PSA names will change from one data center to another. Each PSA search must be surrounded by its own <granuleCondition> tag. Each value must be a single-quoted string.

Code Example 79: PSA search element. Search for all granules with a 'COORDINATE_UNITS_NAME' PSA in which that PSA has a value of METERS.

```
<granuleCondition>
 <psa>
  <psaName>'COORDINATE_UNITS_NAME'</psaName>
  <psaValue><value>'METERS'</value></psaValue>
 </psa>
</granuleCondition>
```

Code Example 80: PSA search element. Search for all granules with a 'SAMPLE_DATE' PSA in which that PSA has a date range between 05/12/2000 and 10/12/2000.

```
<granuleCondition>
  <psa>
    <psaName>'SAMPLE_DATE'</psaName>
    <psaValue>
      <dateRange>
        <startDate>
          <Date YYYY="2000" MM="05"  DD="12"/>
        </startDate>
        <stopDate>
          <Date YYYY="2000" MM="10" DD="12"/>
        </stopDate>
      </dateRange>
     </psaValue>
     </psa>
</granuleCondition>
```

3.9.6 Miscellaneous Elements

3.9.6.1 list

This element is used to specify a list of values. The values may be numbers, dates or strings. Each individual value must appear as text of the value element. If a list of strings is specified it must be a single-quoted string with no wild-card characters. The parent element will be searched using Oracle's IN operator for an exact match with any value in the list. The search is case insensitive.

In general, interpretation of the list will be dependent on the parent element in which it appears. Refer to the specific parent element in which the list appears for a detailed explanation.

Code Example 81: List element.

```
      <list>
         <value>'Africa'</value>
         <value>'Bermuda'</value>
         <value>'Indian Ocean'</value>
      </list>
```

3.9.6.2 textPattern

The textPattern element has an operator attribute that specifies what kind of pattern matching algorithm to use. There are two operators for the pattern matching. The LIKE operator as defined in the Oracle manuals and the Oracle InterMedia text search operator. Currently only the LIKE operator has been implemented. For the LIKE operator the string specified to be matched must be a single-quoted string with or without wild-card characters as specified in the Oracle Manuals.

The wild characters supported are '%' and '_' that are placeholders for replace by any number of characters and replace any one character respectively.

Table 3-10: Wild card use.

Oracle Pattern for the LIKE operator	Interpretation
`'%JOHN%'`	Strings containing 'JOHN' as a sub-string
`'MARY%'`	Strings beginning with the characters 'MARY'
`'%BOB'`	Strings ending in 'BOB'
`'D_VE'`	match words like 'DAVE', 'DOVE', etc.

You can include the actual characters "%" or "_" in the pattern by using the ESCAPE character '\'. If the escape character appears in the pattern before the character "%" or "_" then Oracle interprets this character literally in the pattern, rather than as a special pattern matching character.

Table 3-11: Escape character use.

Oracle Pattern for the LIKE operator with the ESCAPE character '\'	Interpretation
`'JOHN\%THAN'`	String 'JOHN%THAN'
`'JOHN%THAN'`	Strings 'JOHNNATHAN', 'JOHNASDTHAN' etc.
`'JOHN\\MARY'`	String 'JOHN\MARY'

If the parent element (<collectionCondition> or <granuleCondition>) has the attribute negated with value 'Y' then the search will look for string NOT like the pattern specified. The search is case insensitive.

Code Example 82: TextPattern element.

```
<spatialKeywords>
  <textPattern  operator="LIKE">'america%'</textPattern>
</spatialKeywords>
```

3.9.6.3 value

The value element is used to specify one value. Refer to each individual parent element for a detailed explanation. In general, if the element represents a String pattern the search will be case insensitive.

Code Example 83: Value search element.

```
<value>12.6</value>

or

<value>'Asx'</value>
```

3.9.6.4 Date

The date element is used to specify a date value.

Code Example 84: Date search element.

```
<Date YYYY="2000" MM="05" DD="03"/>

or

<Date YYYY="2000" MM="10" DD="09" HH="21" MI="04" SS="32"/>
```

3.9.6.5 dateRange

The dateRange element is used to specify a range of time. It can include just a startDate, just a stopDate or both. If there is just a startDate, then it is declaring any time from that date forward inclusively. If there is just a stopDate, then it is declaring any time before that date inclusive. If it includes both a startDate and a stopDate then it is declaring any time between those two dates inclusive.

Code Example 85: DateRange search element.

```
<dateRange>
   <startDate><Date YYYY="2000" MM="05" DD="03"/></startDate>
   <stopDate><Date YYYY="2000" MM="10" DD="09"/></stopDate>
</dateRange>

or

<dateRange>
   <stopDate>
      <Date YYYY="2000" MM="10" DD="09" HH="04" MI="12" SS="34"/>
   </stopDate>
</dateRange>
```

3.9.7 Present Using GetMetadata API Call

Users have flexibility of getting metadata information without providing resultSetID that was obtained from previous Query API call. Instead, by specifying the ECHOItemIDs they are interested in directly, users can get the metadata information through GetMetadata API call. The parameters that are required to perform the GetMetadata are ItemId, as well as MessageFormat and PresentationDescription, which have been described in the Present API section. One restriction in using this transaction is that users have to make sure all the item ids must have the same data type, either COLLECION or GRANULE, otherwise an exception will be thrown to state which item id has problem with the data type.

Code Example 86: GetMetadata API call.

```
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <GetMetadataRequest>
     <ItemID>C10800-ORNL</ItemID>
     <ItemID>C10802-ORNL</ItemID>
    <PresentationDescription>
      <TupleType>
```

```
        <attributeName>Sensor</attributeName>
        <PrimitiveTypeName>
           <String/>
        </PrimitiveTypeName>
     </TupleType>
     <TupleType>
        <attributeName>Campaign</attributeName>
        <PrimitiveTypeName>
        <String/>
     </PrimitiveTypeName>
      </TupleType>
         <DTDType>
            <BMGT/>
         </DTDType>
   </PresentationDescription>
   </GetMetadataRequest>
</CatalogService>
```

Code Example 87: Request response. Note that this example is based on the first version of BMGT.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <GetMetadataResponse>
    <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED/>
      </BooleanResultType>
    </BooleanResult>
    <ReturnData>
    <MessageFormat>
      <XML/>
    </MessageFormat>
    <payload>
      <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "http://api.echo.eos.nasa.gov/echo/dtd/BMGTCollectionResults.dtd">
<results>
  <provider name='ORNL-DAAC'>
    <result itemId='C10800-ORNL' number='1'>
      <CollectionMetaData>
        <ECHOItemId>C10800-ORNL</ECHOItemId>
        <DataSetId>BOREAS AES FIVE-DAY AVERAGED SURFACE METEOROLOGICAL AND UPPER AIR DATA</DataSetId>
        <Platform>
          <PlatformShortName>METEOROLOGICAL STATION</PlatformShortName>
          <Instrument>
            <InstrumentShortName>HUMAN OBSERVER</InstrumentShortName>
            <Sensor>
              <SensorShortName>HUMAN OBSERVER</SensorShortName>
              <SensorLongName>HUMAN OBSERVER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>ANEMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>ANEMOMETER</SensorShortName>
              <SensorLongName>ANEMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>BAROMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>BAROMETER</SensorShortName>
              <SensorLongName>BAROMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>DEWPOINT HYDROMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>DEWPOINT HYDROMETER</SensorShortName>
              <SensorLongName>DEWPOINT HYDROMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>DRY BULB THERMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>DRY BULB THERMOMETER</SensorShortName>
              <SensorLongName>DRY BULB THERMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>RAIN GAUGE</InstrumentShortName>
            <Sensor>
              <SensorShortName>RAIN GAUGE</SensorShortName>
              <SensorLongName>RAIN GAUGE</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>SNOW MEASURING ROD</InstrumentShortName>
            <Sensor>
              <SensorShortName>SNOW MEASURING ROD</SensorShortName>
              <SensorLongName>SNOW MEASURING ROD</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>WET BULB THERMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>WET BULB THERMOMETER</SensorShortName>
              <SensorLongName>WET BULB THERMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
        </Platform>
        <Campaign>
          <CampaignShortName>BOREAS</CampaignShortName>
          <CampaignLongName>BOREAS</CampaignLongName>
```

```
      </Campaign>
    </CollectionMetaData>
  </result>
  <result itemId='C10802-ORNL' number='2'>
    <CollectionMetaData>
      <ECHOItemId>C10802-ORNL</ECHOItemId>
      <DataSetId>BOREAS AES CANADIAN HOURLY AND DAILY SURFACE METEOROLOGICAL DATA</DataSetId>
      <Platform>
        <PlatformShortName>METEOROLOGICAL STATION</PlatformShortName>
        <Instrument>
          <InstrumentShortName>HUMAN OBSERVER</InstrumentShortName>
          <Sensor>
            <SensorShortName>HUMAN OBSERVER</SensorShortName>
            <SensorLongName>HUMAN OBSERVER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>ANEMOMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>ANEMOMETER</SensorShortName>
            <SensorLongName>ANEMOMETER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>BAROMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>BAROMETER</SensorShortName>
            <SensorLongName>BAROMETER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>DEWPOINT HYDROMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>DEWPOINT HYDROMETER</SensorShortName>
            <SensorLongName>DEWPOINT HYDROMETER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>DRY BULB THERMOMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>DRY BULB THERMOMETER</SensorShortName>
            <SensorLongName>DRY BULB THERMOMETER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>RAIN GAUGE</InstrumentShortName>
          <Sensor>
            <SensorShortName>RAIN GAUGE</SensorShortName>
            <SensorLongName>RAIN GAUGE</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>SNOW MEASURING ROD</InstrumentShortName>
          <Sensor>
            <SensorShortName>SNOW MEASURING ROD</SensorShortName>
            <SensorLongName>SNOW MEASURING ROD</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>WET BULB THERMOMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>WET BULB THERMOMETER</SensorShortName>
            <SensorLongName>WET BULB THERMOMETER</SensorLongName>
          </Sensor>
        </Instrument>
      </Platform>
      <Campaign>
        <CampaignShortName>BOREAS</CampaignShortName>
        <CampaignLongName>BOREAS</CampaignLongName>
      </Campaign>
    </CollectionMetaData>
  </result>
</provider>
</results>]]>
      </payload>
    </ReturnData>
  </GetMetadataResponse>
</CatalogService>
```

3.10 Visibility of Results

When a user executes a Query transaction, the query is applied to all the data in ECHO to create the result set. But when the results are to be presented, not all results are visible to the user. The visibility of the data depends on the rules defined by the Provider of the data and the privileges that the user has. Please refer to ECHO Requirements on Visibility for more details.

If a particular result is not visible to the user, the result will specify the ECHO id for that result and specify that it is not visible. The user may later inquire with ECHO what steps should be taken in order for the results to be visible if at all possible.

Code Example 88: Results not visible to the user.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "http://api.echo.eos.nasa.gov/echo/dtd/ECHOGranuleResults.dtd">
<results>
  <provider name='ORNL-DAAC'>
    <result number='1'>G70958-ORNL-DAAC is not visible</result>
    <result number='2'>G70959-ORNL-DAAC is not visible</result>
  </provider>
</results>
```

3.11 Deleted Results

When a user executes a Query, all the data that conforms to the Query are saved as the result set of the Query. This result set is saved within ECHO. The user can come back anytime later to view this result set (provided it has not already been deleted from ECHO). It is possible that between the time the Query is executed and the results are presented, some of the data is deleted from ECHO due to a request from the Provider. In that case the result will specify the ECHO id for that result and specify it as deleted.

The DTD for the result (as noted in the XML itself) is located at:

http://api.echo.eos.nasa.gov/echo/dtd/ECHOGranuleResults.dtd

The root element, <results> begins every result XML document. The results are grouped by provider. Hence, the element under result is provider with an attribute name that specifies the name of the provider.

3.12 Sample Queries

Please note that some of the sample queries may result in 0 hits.

3.12.1 Queries for Collections (Discovery Search)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for collections from ORNL that have
  parameter value that contains 'IMAGERY'
-->
<query>
 <for value="collections"/>
 <dataCenterId>
   <list><value>ORNL-DAAC</value></list>
 </dataCenterId>
 <where>
    <collectionCondition>
      <parameter>
        <textPattern>'%Imagery%'</textPattern>
      </parameter>
    </collectionCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
   Search for collections from GSFCECS and ORNL that have
   processing level 1A or 2
-->
<query>
 <for value="collections"/>
  <dataCenterId>
      <list>
         <value>GSFCECS</value>
         <value>ORNL-DAAC</value>
      </list>
  </dataCenterId>
 <where>
  <collectionCondition negated="y">
    <processingLevel><list>
         <value>'1A'</value>
```

```
        <value>'2'</value>
        </list></processingLevel>
  </collectionCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
   Search for collections from ORNL with:
   temporal range: periodic range between Jan 1, 1990 and Dec. 31 1998
                 from the 1st to the 300th day of each year, AND
   spatial extent: bounding box 60S, 70W to 60N, 70E.
-->
<query>
 <for value="collections"/>
  <dataCenterId>
     <value>ORNL-DAAC</value>
  </dataCenterId>
 <where>
  <collectionCondition>
   <temporal>
      <startDate><Date YYYY="1990" MM="01" DD="01"/></startDate>
      <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
      <startDay value="1"/>
      <endDay value="300"/>
   </temporal>
  </collectionCondition>
  <collectionCondition negated="n">
    <spatial operator="RELATE">
       <IIMSPolygon>
         <IIMSLRing>
           <IIMSPoint  long='-10' lat='85'/>
           <IIMSPoint  long='10' lat='85'/>
           <IIMSPoint  long='10' lat='89'/>
           <IIMSPoint  long='-10' lat='89'/>
           <IIMSPoint  long='-10' lat='85'/>
         </IIMSLRing>
       </IIMSPolygon>   </spatial>
  </collectionCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for collections from ORNL with
temporal range: periodic range between Jan 1, 1990 and Dec. 31 1998
                from the 1st to the 300th day of each year, AND
                some days of January.
source name: L7 or AM-1 AND
spatially covering any 'temperate' region or USA
```

```
-->
<query>
 <for value="collections"/>
  <dataCenterId>
     <list><value>ORNL-DAAC</value></list>
  </dataCenterId>
 <where>
  <collectionCondition>
   <temporal>
      <startDate><Date YYYY="1990" MM="01" DD="01"/></startDate>
      <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
      <startDay value="1"/>
      <endDay value="300"/>
   </temporal>
  </collectionCondition>
  <collectionCondition negated='n'>
     <sourceName>
        <list>
           <value>'L7'</value>
           <value>'AM-1'</value>
        </list>
     </sourceName>
  </collectionCondition>

  <collectionCondition>
    <spatialKeywords>
        <list>
           <value>'temperate'</value>
           <value>'USA'</value>
        </list>
    </spatialKeywords>
  </collectionCondition>

  <collectionCondition>
    <temporalKeywords>
       <textPattern>'%january%'</textPattern>
    </temporalKeywords>
  </collectionCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
        <for value="collections"/>
        <dataCenterId>
                <value>ORNL-DAAC</value>
        </dataCenterId>
        <where>
           <collectionCondition>
                <psaNames>
               <list>
                     <value>'Provider_Specific_Attribute_1'</value>
                 <value>'Provider_Specific_Attribute_3'</value>
               </list>
                </psaNames>
           </collectionCondition>
        </where>
</query>
```

3.12.2 Queries for Granules (Inventory Search)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules from the L70R or L70RWRS or GLCF_GRANULE_METADATA datasets that have browse data and were categorized
as Day granules.
-->
<query>
 <for value="granules"/>
  <dataCenterId>
     <list><value>ORNL-DAAC</value></list>
  </dataCenterId>
 <where>
 <granuleCondition><browseOnly/>
 </granuleCondition>

 <granuleCondition>
    <cloudCover>
       <range lower="10" upper="20"/>
    </cloudCover>
</granuleCondition>

 <granuleCondition><dataSetId>
    <list>
       <value>'L70R'</value>
       <value>'L70RWRS'</value>
       <value>'GLCF_GRANULE_METADATA'</value>
    </list>
 </dataSetId></granuleCondition>

 <granuleCondition><dayNightFlag value="DAY"/>
 </granuleCondition>

 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules the specified spatial region, and periodic temporal extent
-->
<query>
 <for value="granules"/>
```

```
  <dataCenterId>
     <list><value>ORNL-DAAC</value></list>
  </dataCenterId>
 <where>
  <granuleCondition>
   <spatial operator="RELATE">
     <Polygon>
       <LRing>
         <CList>-120, -30, -100, -60, 5, -90, 5, 85, -120, 30, -120, -30</CList>
       </LRing>
       <LRing>
         <CList>80, 20, 80, 60, 20, 60, 20, 20, 80, 20</CList>
       </LRing>
     </Polygon>
   </spatial>
 </granuleCondition>

 <granuleCondition>
   <temporal>
        <startDate><Date YYYY="1980" MM="01" DD="01"/></startDate>
        <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
        <startDay value="1"/>
        <endDay value="300"/>
   </temporal>
 </granuleCondition>

 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules from ORNL with source name SWIR or TIR
-->
<query>
 <for value="granules"/>
 <dataCenterId>
   <list><value>ORNL-DAAC</value></list>
 </dataCenterId>
 <where>
 <granuleCondition>
   <sensorName><list>
         <value>'SWIR'</value>
         <value>'TIR'</value></list>
   </sensorName>
 </granuleCondition>

 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
```

```
<!--
Search for granules with temporal range: periodic range between Jan 1, 1990 and Dec. 31 1998 from the 1st to the 300th day
of each year.
-->
<query>
 <for value="granules"/>
 <dataCenterId><list><value>ORNL-DAAC</value></list>
 </dataCenterId>
 <where>
 <granuleCondition>
   <temporal>
       <startDate><Date YYYY="1990" MM="01" DD="01"/></startDate>
       <stopDate><Date YYYY="1998" MM="12" DD="31"/></stopDate>
       <startDay value="1"/>
       <endDay value="300"/>
   </temporal>
 </granuleCondition>
 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<!--
Search for granules in the specified list of granule ID s AND in the project/campaign 'rivers' and with PATH=15 and ROW =
33 for WRS1 type.
-->
<query>
 <for value="granules"/>
  <dataCenterId>
     <list><value>ORNL-DAAC</value><value>GSFCECS</value></list>
  </dataCenterId>
 <where>
 <granuleCondition>
     <CampaignShortName><list>
        <value>'rivers'</value>
     </list></CampaignShortName>
 </granuleCondition>

 <granuleCondition><ECHOGranuleID>
    <list>
       <value>'G_ORNL_941.1'</value>
       <value>'G_ORNL_937.1'</value>
       <value>'G_ORNL_939.1'</value>
       <value>'G_ORNL_768.1'</value>
       <value>'ECS2:GR:2000059170'</value>
       <value>'ECS2:GR:2000055252'</value>
       <value>'ECS2:GR:2000058193'</value>
       <value>'ECS2:GR:2000059012'</value>
       <value>'ECS2:GR:2000058825'</value>
       <value>'ECS2:GR:2000059048'</value>
    </list>
 </ECHOGranuleID></granuleCondition>
 <granuleCondition>
    <pathRow>
        <path><range lower='15' upper='15'/></path>
        <row><range lower='33' upper='33'/></row>
        <wrsType value='1'/>
    </pathRow>
 </granuleCondition>

 </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
        <for value="granules"/>
        <dataCenterId>
                <value>ORNL-DAAC</value>
        </dataCenterId>
        <where>
           <granuleCondition>
                <psa>
                   <psaName>'COORDINATE_UNITS_NAME'</psaName>
                   <psaValue><value>'METERS'</value></psaValue>
                </psa>
           </granuleCondition>
        </where>
</query>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
   <for value="granules"/>
   <dataCenterId>
      <value>ORNL-DAAC</value>
   </dataCenterId>
   <where>
      <granuleCondition>
         <psa>
            <psaName>'SAMPLE_DATE'</psaName>
               <psaValue>
                  <dateRange>
                     <startDate>
                       <Date YYYY="2000" MM="05" DD="12"/></startDate>
                     <stopDate>
                       <Date YYYY="2000" MM="10" DD="12"/>
                     </stopDate>
                  </dateRange>
               </psaValue>
             </psa>
      </granuleCondition>
    </where>
</query>
```

3.12.3 QueryRequest

Below is an example QueryRequest with nothing specified for PresentationDescription (taking default values) and its response with results. Note that the actual result string is embedded in a wrapper XML message within a CDATA field, so that even though it looks like XML, it will have to be extracted in order to parse it as XML, which is done to accommodate non-XML return formats.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <QueryRequest>
      <QueryExpression>
         <query>
<![CDATA[

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE query PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/IIMSAQLQueryLanguage.dtd">
<query>
 <for value="granules"/>
  <dataCenterId>
     <value>ORNL-DAAC</value>
  </dataCenterId>
 <where>
  <granuleCondition>
     <dataSetId><textPattern>'%Global%'</textPattern></dataSetId>
  </granuleCondition>
 </where>
</query>

]]>
         </query>
         <namespace>none</namespace>
         <QueryLanguage>
            <IIMSAQL/>
         </QueryLanguage>
      </QueryExpression>
      <ResultType>
         <RESULTS/>
      </ResultType>
            <IteratorSize>2</IteratorSize>
   </QueryRequest>
</CatalogService>
```

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <QueryResponse>
    <BooleanResult>
      <BooleanResultType>
```

```
        <REQUEST_SUCCEEDED/>
      </BooleanResultType>
    </BooleanResult>
    <ReturnData>
    <MessageFormat>
      <XML/>
    </MessageFormat>
    <payload>
      <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "http://api.echo.eos.nasa.gov/echo/dtd/ECHOGranuleResults.dtd">
<results>
  <provider name='ORNL-DAAC'>
    <result number='1' itemId='G2530-ORNL'>
      <GranuleURMetaData>
        <ECHOItemId>G2530-ORNL</ECHOItemId>
        <GranuleUR>DUNNESOIL.soilwhc.dat</GranuleUR>
        <ECHOInsertDate>2002-04-10 14:56:40.0</ECHOInsertDate>
        <ECHOLastUpdate>2002-03-20 14:56:44.0</ECHOLastUpdate>
        <GranuleShortName>soilwhc.dat</GranuleShortName>
        <AccessConstraint>NOT_ACCESS_RESTRICTED</AccessConstraint>
        <Price>0</Price>
        <CatalogItemId>G2530-ORNL</CatalogItemId>
        <CollectionMetaData>
          <ShortName>GLOBAL DISTRIBUTION OF PLANT-EXTRACTABLE WATER CAPACITY OF SOIL (DUNNE)</ShortName>
          <VersionID>0</VersionID>
          <DataSetId>GLOBAL DISTRIBUTION OF PLANT-EXTRACTABLE WATER CAPACITY OF SOIL (DUNNE)</DataSetId>
        </CollectionMetaData>
        <DataGranule>
          <SizeMBDataGranule>73715</SizeMBDataGranule>
          <ProductionDateTime>1980-06-01 00:00:00.0</ProductionDateTime>
        </DataGranule>
        <RangeDateTime>
          <RangeEndingTime>00:00:00</RangeEndingTime>
          <RangeEndingDate>1996-12-31 00:00:00.0</RangeEndingDate>
          <RangeBeginningTime>00:00:00</RangeBeginningTime>
          <RangeBeginningDate>1996-01-01 00:00:00.0</RangeBeginningDate>
        </RangeDateTime>
        <SpatialDomainContainer>
          <HorizontalSpatialDomainContainer>
<BoundingRectangle><WestBoundingCoordinate>-
180</WestBoundingCoordinate><NorthBoundingCoordinate>90</NorthBoundingCoordinate><EastBoundingCoordinate>180</EastBoundingC
oordinate><SouthBoundingCoordinate>-90</SouthBoundingCoordinate></BoundingRectangle>

          </HorizontalSpatialDomainContainer>
        </SpatialDomainContainer>
        <MeasuredParameter>
          <MeasuredParameterContainer>
            <ParameterName>SOIL MOISTURE/WATER CONTENT SOIL WATER HOLDING CAPACITY</ParameterName>
          </MeasuredParameterContainer>
        </MeasuredParameter>
        <StorageMediumClass>
          <StorageMedium>On-Line</StorageMedium>
        </StorageMediumClass>
        <Platform>
          <PlatformShortName>TOPOGRAPHIC MAP</PlatformShortName>
          <Instrument>
            <InstrumentShortName>ANALYSIS</InstrumentShortName>
            <Sensor>
              <SensorShortName>ANALYSIS</SensorShortName>
            </Sensor>
          </Instrument>
        </Platform>
        <Contact>
          <Role>User Services Office</Role>
          <ContactOrganizationName>ORNL DAAC User Services Office</ContactOrganizationName>
          <ContactOrganizationAddress>
            <StreetAddress>Oak Ridge National Laboratory,      P.O. Box 2008, MS 6407</StreetAddress>
            <City>Oak Ridge</City>
            <StateProvince>Tennessee</StateProvince>
            <PostalCode>37831-6407</PostalCode>
            <Country>USA</Country>
          </ContactOrganizationAddress>
          <ContactOrganizationAddress>
            <StreetAddress>Oak Ridge National Laboratory,P.O. Box 2008, MS 6407</StreetAddress>
            <City>Oak Ridge</City>
            <StateProvince>Tennessee</StateProvince>
            <PostalCode>37831-6407</PostalCode>
            <Country>USA</Country>
          </ContactOrganizationAddress>
          <OrganizationTelephone>
            <TelephoneNumber>+1 (865) 241-3952</TelephoneNumber>
            <TelephoneType>Telephone</TelephoneType>
          </OrganizationTelephone>
          <OrganizationEmail>
            <ElectronicMailAddress>ornldaac@ornl.gov</ElectronicMailAddress>
          </OrganizationEmail>
        </Contact>
        <OnlineAccessURLs>
          <OnlineAccessURL>
            <URL>http://daac.ornl.gov/global_soil/Dunne/data/soilwhc.dat</URL>
            <MimeType>text</MimeType>
          </OnlineAccessURL>
        </OnlineAccessURLs>
      </GranuleURMetaData>
    </result>
  </provider>
</results>
]]>
      </payload>
    </ReturnData>
    <RequestID>RGuest7213396921014333628487</RequestID>
    <ResultSetID>RGuest7213396921014333628487</ResultSetID>
    <ResultType>
      <RESULTS/>
    </ResultType>
    <Status>
```

```
      <SUCCESS_RESULTS_AVAILABLE/>
    </Status>
    <Hits>1</Hits>
    <Cursor>2</Cursor>
  </QueryResponse>
</CatalogService>
```

3.12.4 PresentRequest – Example 1

The following is an example of a PresentRequest with no ResultSet specified and only Sensor and Campaign TupleTypes for BMGT format Collection results. The presented results are shown based on the first version of BMGT.

Code Example 89: Present request 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <PresentRequest>
     <ResultSetID>RU10021060284589150</ResultSetID>
     <PresentationDescription>
      <TupleType>
         <attributeName>Sensor</attributeName>
         <PrimitiveTypeName>
            <String/>
         </PrimitiveTypeName>
      </TupleType>
      <TupleType>
         <attributeName>Campaign</attributeName>
         <PrimitiveTypeName>
            <String/>
          </PrimitiveTypeName>
       </TupleType>
       <DTDType>
          <BMGT/>
       </DTDType>
     </PresentationDescription>
    <IteratorSize>2</IteratorSize>
     <Cursor>1</Cursor>
   </PresentRequest>
</CatalogService>
```

Code Example 90: Present response 1.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE PUBLIC "-//ECHO CatalogService (v5.5)//EN" "http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <PresentResponse>
    <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED/>
      </BooleanResultType>
    </BooleanResult>
    <ReturnData>
    <MessageFormat>
      <XML/>
    </MessageFormat>
    <payload>
      <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "http://api.echo.eos.nasa.gov/echo/dtd/BMGTCollectionResults.dtd">
<results>
  <provider name='ORNL-DAAC'>
    <result number='1' itemId='C10800-ORNL-DAAC'>
      <CollectionMetaData>
        <ECHOItemId>C10800-ORNL-DAAC</ECHOItemId>
        <DataSetId>BOREAS AES FIVE-DAY AVERAGED SURFACE METEOROLOGICAL AND UPPER AIR DATA</DataSetId>
        <Platform>
          <PlatformShortName>METEOROLOGICAL STATION</PlatformShortName>
          <Instrument>
            <InstrumentShortName>HUMAN OBSERVER</InstrumentShortName>
            <Sensor>
              <SensorShortName>HUMAN OBSERVER</SensorShortName>
              <SensorLongName>HUMAN OBSERVER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>ANEMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>ANEMOMETER</SensorShortName>
              <SensorLongName>ANEMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>BAROMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>BAROMETER</SensorShortName>
              <SensorLongName>BAROMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>DEWPOINT HYDROMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>DEWPOINT HYDROMETER</SensorShortName>
              <SensorLongName>DEWPOINT HYDROMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>DRY BULB THERMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>DRY BULB THERMOMETER</SensorShortName>
              <SensorLongName>DRY BULB THERMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
        <Instrument>
          <InstrumentShortName>RAIN GAUGE</InstrumentShortName>
          <Sensor>
            <SensorShortName>RAIN GAUGE</SensorShortName>
            <SensorLongName>RAIN GAUGE</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>SNOW MEASURING ROD</InstrumentShortName>
          <Sensor>
            <SensorShortName>SNOW MEASURING ROD</SensorShortName>
            <SensorLongName>SNOW MEASURING ROD</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>WET BULB THERMOMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>WET BULB THERMOMETER</SensorShortName>
            <SensorLongName>WET BULB THERMOMETER</SensorLongName>
          </Sensor>
        </Instrument>
      </Platform>
      <Campaign>
        <CampaignShortName>BOREAS</CampaignShortName>
        <CampaignLongName>BOREAS</CampaignLongName>
      </Campaign>
    </CollectionMetaData>
  </result>
  <result number='2' itemId='C10802-ORNL-DAAC'>
    <CollectionMetaData>
      <ECHOItemId>C10802-ORNL-DAAC</ECHOItemId>
      <DataSetId>BOREAS AES CANADIAN HOURLY AND DAILY SURFACE METEOROLOGICAL DATA</DataSetId>
      <Platform>
        <PlatformShortName>METEOROLOGICAL STATION</PlatformShortName>
        <Instrument>
          <InstrumentShortName>HUMAN OBSERVER</InstrumentShortName>
          <Sensor>
            <SensorShortName>HUMAN OBSERVER</SensorShortName>
            <SensorLongName>HUMAN OBSERVER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>ANEMOMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>ANEMOMETER</SensorShortName>
            <SensorLongName>ANEMOMETER</SensorLongName>
          </Sensor>
        </Instrument>
        <Instrument>
          <InstrumentShortName>BAROMETER</InstrumentShortName>
          <Sensor>
            <SensorShortName>BAROMETER</SensorShortName>
            <SensorLongName>BAROMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>DEWPOINT HYDROMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>DEWPOINT HYDROMETER</SensorShortName>
              <SensorLongName>DEWPOINT HYDROMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>DRY BULB THERMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>DRY BULB THERMOMETER</SensorShortName>
              <SensorLongName>DRY BULB THERMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>RAIN GAUGE</InstrumentShortName>
            <Sensor>
              <SensorShortName>RAIN GAUGE</SensorShortName>
              <SensorLongName>RAIN GAUGE</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>SNOW MEASURING ROD</InstrumentShortName>
            <Sensor>
              <SensorShortName>SNOW MEASURING ROD</SensorShortName>
              <SensorLongName>SNOW MEASURING ROD</SensorLongName>
            </Sensor>
          </Instrument>
          <Instrument>
            <InstrumentShortName>WET BULB THERMOMETER</InstrumentShortName>
            <Sensor>
              <SensorShortName>WET BULB THERMOMETER</SensorShortName>
              <SensorLongName>WET BULB THERMOMETER</SensorLongName>
            </Sensor>
          </Instrument>
        </Platform>
        <Campaign>
          <CampaignShortName>BOREAS</CampaignShortName>
          <CampaignLongName>BOREAS</CampaignLongName>
        </Campaign>
      </CollectionMetaData>
    </result>
  </provider>
</results>]]>
      </payload>
    </ReturnData>
    <Cursor>3</Cursor>
    <Hits>293</Hits>
    <Status>
      <SUCCESS_RESULTS_AVAILABLE/>
    </Status>
  </PresentResponse>
</CatalogService>
```

3.12.5 Present Request – Example 2

Below is a sample PresentRequest with a previously saved result set returning GranuleVersionId, SizeMBData, Price, shortName and sensor information in ECHO format.

Code Example 91: Present request 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
   <PresentRequest>
   <ResultSetID>result1</ResultSetID>
   <PresentationDescription>
        <TupleType>
         <attributeName>GranuleVersionId</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <TupleType>
         <attributeName>SizeMBDataGranule</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <TupleType>
         <attributeName>Price</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <TupleType>
         <attributeName>shortName</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <TupleType>
         <attributeName>sensor</attributeName>
         <PrimitiveTypeName>
          <String/>
         </PrimitiveTypeName>
        </TupleType>
        <DTDType>
             <ECHO/>
         </DTDType>
       </PresentationDescription>
      <IteratorSize>2</IteratorSize>
     <Cursor>1</Cursor>
```

```
  </PresentRequest>
</CatalogService>
```

Code Example 92: Present response 2.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
<CatalogService>
  <PresentResponse>
    <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED/>
      </BooleanResultType>
    </BooleanResult>
    <ReturnData>
    <MessageFormat>
      <XML/>
    </MessageFormat>
    <payload>
      <![CDATA[
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results PUBLIC "-//ECHO GranuleResults (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ECHOGranuleResults.dtd">
<results>
  <provider name='ORNL-DAAC'>
    <result number='1' itemId='G5432-ORNL-DAAC'>
      <GranuleURMetaData>
      <ECHOItemId>G5432-ORNL-DAAC</ECHOItemId>
<GranuleUR>BOREAS_AES5DAVG.canadian_5_day_avg_daily.dat</GranuleUR>
        <shortName>canadian_5_day_avg_daily.dat</shortName>
        <SizeMBDataGranule>4649911</SizeMBDataGranule>
        <sensor name='HUMAN OBSERVER (1001)'/>
        <sensor name='ANEMOMETER (1010)'/>
        <sensor name='BAROMETER (1018)'/>
        <sensor name='DEWPOINT HYDROMETER (1032)'/>
        <sensor name='DRY BULB THERMOMETER (1037)'/>
        <sensor name='RAIN GAUGE (1105)'/>
        <sensor name='SNOW MEASURING ROD (1111)'/>
        <sensor name='WET BULB THERMOMETER (1158)'/>
      </GranuleURMetaData>
    </result>
    <result number='2' itemId='G5444-ORNL-DAAC'>
      <GranuleURMetaData>
      <ECHOItemId>G5444-ORNL-DAAC</ECHOItemId>         <granuleUR>BOREAS_AES5DAVG.canadian_5_day_avg_hourly.dat</granuleUR>
        <shortName>canadian_5_day_avg_hourly.dat</shortName>
        <SizeMBDataGranule>27547674</SizeMBDataGranule>
        <sensor name='HUMAN OBSERVER (1001)'/>
        <sensor name='ANEMOMETER (1010)'/>
        <sensor name='BAROMETER (1018)'/>
        <sensor name='DEWPOINT HYDROMETER (1032)'/>
        <sensor name='DRY BULB THERMOMETER (1037)'/>
        <sensor name='RAIN GAUGE (1105)'/>
        <sensor name='SNOW MEASURING ROD (1111)'/>
        <sensor name='WET BULB THERMOMETER (1158)'/>
      </GranuleURMetaData>
    </result>
    <Dictionary name='sensor'>
      <sensor name='HUMAN OBSERVER (1001)'>
        <sensorShortName>HUMAN OBSERVER</sensorShortName>
        <sensorLongName>HUMAN OBSERVER</sensorLongName>
      </sensor>
      <sensor name='ANEMOMETER (1010)'>
        <sensorShortName>ANEMOMETER</sensorShortName>
        <sensorLongName>ANEMOMETER</sensorLongName>
      </sensor>
      <sensor name='BAROMETER (1018)'>
        <sensorShortName>BAROMETER</sensorShortName>
        <sensorLongName>BAROMETER</sensorLongName>
      </sensor>
      <sensor name='DEWPOINT HYDROMETER (1032)'>
        <sensorShortName>DEWPOINT HYDROMETER</sensorShortName>
        <sensorLongName>DEWPOINT HYDROMETER</sensorLongName>
      </sensor>
      <sensor name='DRY BULB THERMOMETER (1037)'>
        <sensorShortName>DRY BULB THERMOMETER</sensorShortName>
        <sensorLongName>DRY BULB THERMOMETER</sensorLongName>
      </sensor>
      <sensor name='RAIN GAUGE (1105)'>
        <sensorShortName>RAIN GAUGE</sensorShortName>
        <sensorLongName>RAIN GAUGE</sensorLongName>
      </sensor>
      <sensor name='SNOW MEASURING ROD (1111)'>
        <sensorShortName>SNOW MEASURING ROD</sensorShortName>
        <sensorLongName>SNOW MEASURING ROD</sensorLongName>
      </sensor>
      <sensor name='WET BULB THERMOMETER (1158)'>
        <sensorShortName>WET BULB THERMOMETER</sensorShortName>
        <sensorLongName>WET BULB THERMOMETER</sensorLongName>
      </sensor>
    </Dictionary>
  </provider>
</results>]]>
    </payload>
    </ReturnData>
    <Cursor>3</Cursor>
    <Hits>821</Hits>
    <Status>
      <SUCCESS_RESULTS_AVAILABLE/>
    </Status>
  </PresentResponse>
</CatalogService>
```

3.13 Provider Profile Service

The provider profile service is for users to retrieve a providers' profile, e.g. the provider's ID in ECHO (particularly useful in query), the organization names of providers, contact information of providers, and transactions supported by providers. Any user could use this service to get the provider's information.

3.14 Transactions

3.14.1 ListAllProviders

This transaction lists all provider IDs in ECHO.

Code Example 93: ListAllProviders transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderProfileService PUBLIC "-//ECHO ProviderProfileService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderProfileService.dtd">
<ProviderProfileService>
   <ListAllProvidersRequest/>
</ProviderProfileService>
```

3.14.2 PresentProviderProfiles

This transaction presents provider's profile, including organization name, spatial projection type, description of holdings provider has, all of provider's contact information.

Code Example 94: PresentProvderProfile transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderProfileService PUBLIC "-//ECHO ProviderProfileService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderProfileService.dtd">
<ProviderProfileService>
   <PresentProviderProfilesRequest>
      <ProviderID>ORNL-DAAC</ProviderID>
   </PresentProviderProfilesRequest>
</ProviderProfileService>
```

3.14.3 PresentProviderSupportedTransactions

This transaction presents the transactions that the provider supports. Currently there are just some order-related transactions (e.g., order submit, order cancel).

Code Example 95: PresentProvderSupportedTransactions transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderProfileService PUBLIC "-//ECHO ProviderProfileService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderProfileService.dtd">
<ProviderProfileService>
   <PresentProviderSupportedTransactionsRequest>
      <ProviderID>ORNL-DAAC</ProviderID>
   </PresentProviderSupportedTransactionsRequest>
</ProviderProfileService>
```

3.15 Group Management Service

The Group Management Service is used to aggregate users together to form groups. Each group created within ECHO has a specific owner. The owner has the option of managing the group or delegating management of the group to one or more other registered users in the system. There is no restriction on the type of user that a manager must be. Thus, a group can be managed by providers or by individual users.

Group managers are the curators of the group and are responsible for maintaining the group's description, name, membership and management roster. A manager has absolute control over all of the previous attributes of the group. As such, a group's management team should be chosen carefully to prevent misuse.

The following conditions also apply:

1. The individual user should be a registered user.
2. The owner of the group needs to add himself as a manager before he can manage the group.

Groups also provide a communication mechanism. Any user in the ECHO system has the capability to contact a group's management team. Similarly, any manager of a group has the capability to contact the group's members. This could be useful for testers to inform their test team when to suspend testing.

The GroupManagementService provides a mechanism for aggregating users together. The groups created from this service are used in Section 6: Data Management Service of the *ECHO Provider Interface Users Guide*.

3.16 Transactions

3.16.1 ListGroups

The ListGroups transaction will display all the groups managed by the invoking user.

Code Example 96: ListGroups transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <ListGroupsRequest/>
</GroupManagementService>
```

3.16.2 ListAllGroups

The ListAllGroups transaction returns a list of all the group names that exist in the system.

Code Example 97: ListAllGroups transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <ListAllGroupsRequest></ListAllGroupsRequest>
</GroupManagementService>
```

3.16.3 PresentGroupInformation

The PresentGroupInformation transaction displays a group's name, description, and its list of managers.

Code Example 98: PresentGroupInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <PresentGroupInformationRequest>
      <GroupName>Group1</GroupName>
   </PresentGroupInformationRequest>
</GroupManagementService>
```

3.16.4 CreateGroup

The example below creates a group named 'MODIS QA' and identifies Tester1 as the manager of the group. It is important to note that group names must be globally unique. Providers that create groups should therefore be judicious in the names they select.

Code Example 99: CreateGroup transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <CreateGroupRequest>
      <GroupInformation>
         <GroupName>MODIS QA</GroupName>
         <description>
            The test team CSC has assembled to test MODIS metadata
         </description>
         <GroupManager>
            <UserName>Tester1</UserName>
         </GroupManager>
      </GroupInformation>
   </CreateGroupRequest>
</GroupManagementService>
```

3.16.5 UpdateGroupInformation

The UpdateGroupInformation transaction can be used to update a group's description as well as its entire management team. Although we provide AddManager and RemoveManager transactions for granular control of the group's management team, the UpdateGroupInformation transaction is used to completely re-write the management team. This might be useful if a group managed by contractors from one company is handed over to another company for management.

Code Example 100: UpdateGroupInformation transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
```

```
<GroupManagementService>
   <UpdateGroupInformationRequest>
      <GroupInformation>
            <GroupName>MODIS QA</GroupName>
            <description>
            The test team GST has assembled to test MODIS metadata
            </description>
         <GroupManager>
            <UserName>GST_Tester1</UserName>
         </GroupManager>
      </GroupInformation>
   </UpdateGroupInformationRequest>
</GroupManagementService>
```

In the above example, the description of the group has changed to indicate a change in contractors handling MODIS metadata testing. Similarly, GST_Tester1 is the exclusive manager of the group. After this transaction is invoked, Tester1 is no longer a manager of the 'MODIS QA' group. If Tester1 attempts to invoke a transaction against 'MODIS QA' that requires management privileges they will be rejected.

3.16.6 RenameGroup

The RenameGroup transaction will rename a group. The system will send an e-mail to all group members about the change if Notify is true.

Code Example 101: RenameGroup transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <RenameGroupRequest>
      <OldGroupName>Group1</OldGroupName>
      <NewGroupName>Group2</NewGroupName>
      <Notify>true</Notify>
   </RenameGroupRequest>
</GroupManagementService>
```

3.16.7 DestroyGroup

This transaction disbands a group. The system will send an e-mail to all group managers and members if Notify is true.

Code Example 102: DestroyGroup transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <DestroyGroupRequest>
    <GroupName>Group1</GroupName>
    <Notify>true</Notify>
  </DestroyGroupRequest>
</GroupManagementService>
```

3.16.8 ListMembers

The ListMembers transaction will list all the members of a group. Managers are not included in the list unless they have been added as group members also.

Code Example 103: ListMembers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <ListMembersRequest>
  <GroupName>Group1</GroupName>
  </ListMembersRequest>
</GroupManagementService>
```

3.16.9 AddMember

The AddMember transaction will add a member to the group. The system will send an e-mail to the newly-added member if Notify is true.

Code Example 104: AddMember transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <AddMemberRequest>
    <GroupName>Group1</GroupName>
    <UserName>billg</UserName>
    <Notify>true</Notify>
  </AddMemberRequest>
</GroupManagementService>
```

3.16.10 ContactMembers

In addition to aggregating users, groups provide a communication mechanism. Any user in the ECHO system has the capability to contact a group's management team (see Section 5.1.16 ContactManagers). Similarly, using the ContactMembers transaction, any manager of a group has the capability to contact the group's members. This could be useful for testers to inform their test team when to suspend testing. The ContactMembers transaction can only be invoked by a group manager and is used to relay important information to the groups membership roster. The only supported MessageFormat is TXT, but HTML formatted messages are planned to be supported in a future release of the software.

Code Example 105: ContactMembers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <ContactMembersRequest>
      <GroupName>MODIS QA</GroupName>
      <GroupMessage>
         <MessageFormat> <TXT/> </MessageFormat>
         <payload>
         Test Team-
         Stop testing on Friday at 5pm and resume testing Monday morning
         at 8am.  We will have a meeting at noon on Monday to discuss progress.
         Have a great weekend!
         - Your Boss
         </payload>
      </GroupMessage>
   </ContactMembersRequest>
</GroupManagementService>
```

3.16.11 IsMember

The IsMember transaction checks if a user is a member of the specified group.

Code Example 106: IsMember transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <IsMemberRequest>
      <GroupName>Group1</GroupName>
      <UserName>billg</UserName>
   </IsMemberRequest>
</GroupManagementService>
```

3.16.12 RemoveMember

This transaction removes a member from a group. The system will send an e-mail to the removed manager if Notify is true.

Code Example 107: RemoveMember transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <RemoveMemberRequest>
      <GroupName>Group1</GroupName>
      <UserName>linust</UserName>
      <Notify>true</Notify>
   </RemoveMemberRequest>
</GroupManagementService>
```

3.16.13 RemoveAllMembers

This transaction removes all members from a group. The list of managers is not affected, so if a member was both a member and a manager, that member will still be a member after this transaction is completed. The system will send an e-mail to all removed members if Notify is true.

Code Example 108: RemoveAllMembers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <RemoveAllMembersRequest>
      <GroupName>Group1</GroupName>
      <Notify>true</Notify>
   </RemoveAllMembersRequest>
</GroupManagementService>
```

3.16.14 ListManagers

The ListManagers transaction lists all the managers of a group.

Code Example 109: ListManagers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <ListManagersRequest>
    <GroupName>Group1</GroupName>
  </ListManagersRequest>
</GroupManagementService>
```

3.16.15 AddManager

This transaction adds a manager to the group. The new manager is not required to be a member of the group. To add a manager to the list of members of a group, use the AddMember transaction (see Section 5.1.9). The system will send an e-mail to the new manager if Notify is true.

Code Example 110: AddManager transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <AddManagerRequest>
      <GroupName>Group1</GroupName>
      <UserName>billg</UserName>
      <Notify>true</Notify>
   </AddManagerRequest>
</GroupManagementService>
```

3.16.16 ContactManagers

This transaction is used to send a message to a group's managers. Unlike the ContactMembers transaction, any user in the ECHO system may invoke the ContactManagers transaction. This could be used by a user who wishes to become a member of a group and would like to open a dialog with the management team of that group.

Code Example 111: ContactManagers transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
  <ContactManagersRequest>
    <GroupName>Group1</GroupName>
    <GroupMessage>
      <MessageFormat>
        <TXT/>
      </MessageFormat>
      <payload>I want to join your group.</payload>
    </GroupMessage>
  </ContactManagersRequest>
</GroupManagementService>
```

3.16.17 RemoveManager

The RemoveManager transaction removes a manager from a group. A manager cannot be removed if that person is the group's only manager.

Code Example 112: RemoveManager transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE GroupManagementService PUBLIC "-//ECHO GroupManagementService (v5.5)// EN"
"http://api.echo.eos.nasa.gov/echo/dtd/GroupManagementService.dtd">
<GroupManagementService>
   <RemoveManagerRequest>
      <GroupName>Group</GroupName>
      <UserName>stinky2</UserName>
      <Notify>true</Notify>
   </RemoveManagerRequest>
</GroupManagementService>
```

3.17 Error Messages

All errors caused by invoking transactions within the Group Management Service result in a rollback condition. That is, if you pass N groups to the CreateGroup transaction, and ECHO fails to create the Nth group, no groups are created. This behavior is consistent with the rest of the ECHO system. The GroupManagementService is available for use by providers and users. Guests are not permitted to invoke any transaction in the service.

Table 12 gives the possible error messages associated with each transaction within the Group Management Service.

Table 3-12: GMS error messages.

Transaction	Error Message	Description
AddManager AddMember ContactManagers ContactMembers DestroyGroup IsMember ListManagers ListMembers PresentGroupInformation RemoveAllMembers RemoveManager RemoveMember RenameGroup UpdateGroupInformation	Group specified does not exist.	The user specified a group that does not exist.
AddManager AddMember RemoveManager RemoveMember RenameGroup UpdateGroupInformation	User' <username>' is not a manager of a group. Only group managers can add other group managers.	The user that invoked the transaction is not a manager of the group.
AddManager AddMember RemoveManager RemoveMember	User specified does not exist.	The user specified a user that does not exist.
AddManager AddMember	User specified is already a manager.	The user specified a user that is already a manager.
ContactManagers ContactMembers	No message was present.	The user specified a blank message.
CreateGroup	Only providers are able to create groups.	The user that invoked the transaction was a regular user (and a provider is the only person permitted to invoke this transaction).
	Manager specified does not exist.	The user specified a Manager that does not exist.
	<The missing item> was not specified.	The user specified a blank name, Cause, or manager username.
	A group with the name <the name> already exists.	The user specified a group name that is already in use.
RenameGroup	The new group name is already in use.	The 'NewGroup' specified already exists.
	Group specified does not exist.	The 'OldGroup' specified does not exist.
UpdateGroupInformation	Group specified does not have a description.	The user specified a blank description.

Transaction	Error Message	Description
	The user specified does not exist.	The user specified manager(s) that do not exist.
DestroyGroup	Only managers of a group are permitted to destroy the group (and you are not a manager of this group).	The user invoking the transaction is not a manager of the group.
IsMember	User does not exist in the system.	The user specified a user that does not exist.
ListMembers	Only managers of a group are permitted to list the group’s members (and you are not a manager of this group).	The user invoking the transaction is not a manager of the group.
RemoveAllMembers	You are not a manager of the group.	The user invoking the transaction is not a manager of the group.
RemoveManager	User specified is not a manager.	The user specified a user that is not a manager.

3.18 Order Entry Service

The first step in the order process is creating the order. Though it can look like a very simple transaction, a user needs to understand the parts that make up an order (see Figure 3-6).

An order is a collection of order line items. Each order line item consists of a catalog item ID, the quantity of that catalog item, and all the options associated with that catalog item. A single order may consist of many catalog items that belong to many different providers. Each provider may have its own validation scheme and rules for creating, validating and submitting their particular catalog item. To ensure that appropriate validation happens – as well as to demarcate the larger order for sending to each provider – orders can also be split up into smaller provider orders, sometimes called sub-orders. Each provider order is nothing more than all the order line items in a particular order that belong to a specific provider. A provider order is uniquely identified by a provider order ID. A provider order ID is the aggregation of the actual provider ID of the provider in question, and the order ID of the order that contains the catalog items for this provider.

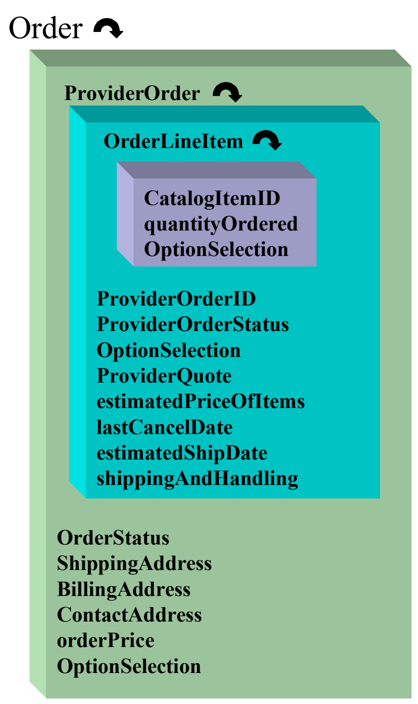


Figure 3-6. Contents of an Order

The most common way to obtain the catalog items needed for creating an order is through the query transactions found in Catalog Service. ECHO uses its own query language called AQL to search for different catalog items. To learn more about the Catalog Service and the actual Query transaction, please refer to Section 3: Catalog Service. To obtain the catalog items, define a query in AQL and pass that query to a transaction like the Query transaction in the Catalog Service (using the QueryExpression parameter). You can also specify in these query transactions exactly how the results should be displayed as well. In the displaying query results, make sure that the 'CatalogItemID' tag is displayed. The string value under this tag is the actual catalog item ID that can be used to order things from the system.

Once the list of catalog items for ordering is obtained, one needs to also know the available or required options for each catalog item that is to be ordered. To list the possible options both allowed and required for each catalog item, use the PresentCatalogItem transaction in the OrderEntryService. After you identify the catalog item ID in the transaction, ECHO will return the complete list of possible options (both required and optional) that one can select for this catalog item.

Note: Sometimes a catalog item may not be orderable (for more information, please refer to the section Restriction on order items and Deleted items covered later in the OrderEntryService). If this is the case, only the catalog item ID and an ErrorMessage is displayed for this item.

Code Example 113: PresentCatalogItem transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentCatalogItemRequest>
      <CatalogItemID>G2076-ORNL</CatalogItemID>
   </PresentCatalogItemRequest>
</OrderEntryService>
```

The option hierarchy in ECHO was created to be as flexible as possible in terms of being able to deal with any kind of option. Because of this flexibility, it can also be a little hard to create the right option selection hierarchy for each catalog item. Initially, one needs to run the PresentCatalogItem transaction, and from that obtain each possible OptionDefinition that it produces. From the OptionDefinition, one needs to make sure that the option selection created for that catalog item follow the option hierarchy that was defined in that transaction. It is also necessary to keep track of all the possible sub-options and sub-sub-options possible for each option. To actually create the option selections for a catalog item, take all the fields in the necessary option definitions (specifically the OptionName element) and map them into the similar hierarchy found in the OptionSelection element, filling in the appropriate Value elements when necessary. All of the information necessary to create an order is complete once the OptionSelection elements for each catalog item in question is filled.

Note: It is not always required to attach options to a certain catalog item. However, if there are any options that are required by that catalog item, they will have to be filled out before one validates and/or submits the order (or else an error will be returned). To learn more about options and how to use them throughout ECHO, refer to the appendices of this document.

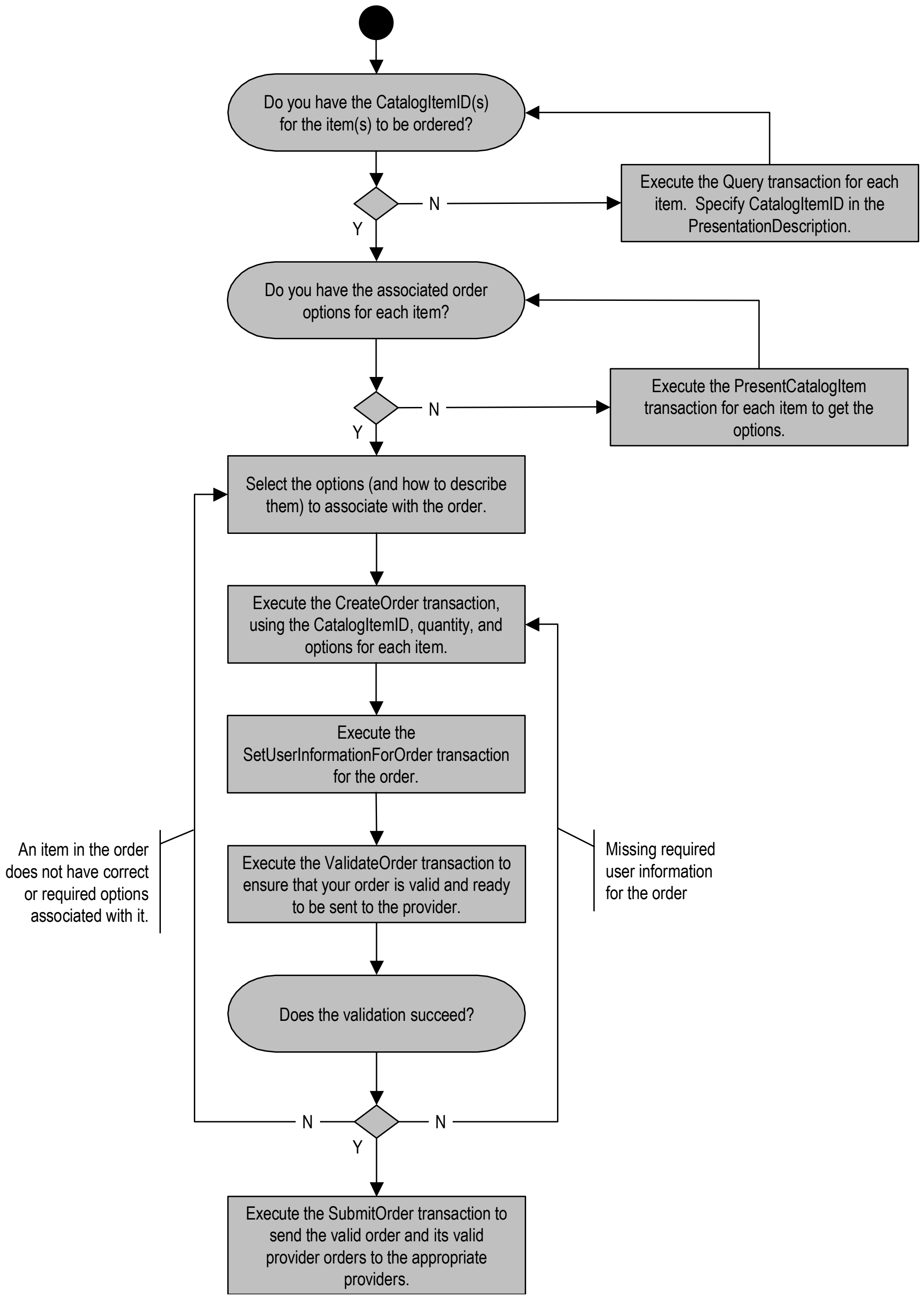
One of ECHO's main tasks is to act as a full functioning order creation and tracking server. In designing the system, flexibility and completeness were top priorities. In keeping these goals in mind, the following aspects of the ordering service were enabled:

- Support for ordering catalog items from a variety of different providers.
- Ability to modify and change a variety of options that may exist for an order, provider-order, and/or catalog item.

Note: An order may be modified as soon as it is created, but modifications are not permitted after the order has been submitted.

Allowing asynchronous flexibility in how and when the user creates a valid and complete order (i.e., the user is not mandated to fill out a complete order in one sitting).

However, in obtaining these goals, there is some complexity involved in fully creating and submitting an order. Though the system allows flexibility in how an order is made, a general workflow for creating, validating, and submitting an order is needed and useful in simplifying the process. Figure 3-7 illustrates the most commonly used and direct approach to creating and submitting an order.



Do you have the CatalogItemID(s) for the item(s) to be ordered?
N
Y
Execute the Query transaction for each item. Specify CatalogItemID in the PresentationDescription.
Do you have the associated order options for each item?
N
Y
Execute the PresentCatalogItem transaction for each item to get the options.
Select the options (and how to describe them) to associate with the order.
Execute the CreateOrder transaction, using the CatalogItemID, quantity, and options for each item.
Execute the SetUserInformationForOrder transaction for the order.
An item in the order does not have correct or required options associated with it.
Execute the ValidateOrder transaction to ensure that your order is valid and ready to be sent to the provider.
Missing required user information for the order
Does the validation succeed?
N
N
Y
Execute the SubmitOrder transaction to send the valid order and its valid provider orders to the appropriate providers.

Figure 3-7. Creating and Submitting an Order

3.19 Establishing a valid client connection

All clients interfacing with the system are required to pass client identifier information to ECHO. ECHO will then take this identifier and attach it to any order submitted through the respective client.

NOTE: If client identifier information is not provided, submitted orders will fail.

The client identifier is a short description and/or name of the client. Client developers are encouraged to keep this information as concise as possible, and to work with the ECHO Operations Group to create an appropriate identifier

The following code example can be used to to create and pass client identifier information. After creating an instance of echoToken, the client can then connect to the desired system (soap URL), e.g. api.echo.eos.nasa.gov or other system, and call the **identify** method to pass the client identifier information.

```
_echoToken = new EchoToken();
_echoToken.connectTo(soapURL);

try
{
  _echoToken.identify("Client identifier");
}
catch (XMLServiceException e)
{
  Logger.logException(e, SoapServer.class, Logger.ERROR);
}
```

The client should execute this code immediately after an ECHO session has been established. This issue must be addressed by the client developer and is essential to the proper submittal of a user order.

3.20 Transactions

3.20.1 CreateOrder

This transaction initiates the order process. Below is an example of an order without options:

Code Example 114: Order without options.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <CreateOrderRequest>
      <OrderLineItem>
         <CatalogItemID>
            2000054170
         </CatalogItemID>
         <quantityOrdered>
            2
         </quantityOrdered>
      </OrderLineItem>
   </CreateOrderRequest>
</OrderEntryService>
```

To create an order with associated options, here is another example:

Code Example 115: Order with options.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <CreateOrderRequest>
      <OrderLineItem>
         <CatalogItemID>
            730
         </CatalogItemID>
         <quantityOrdered>
            2
         </quantityOrdered>
         <OptionSelection>
            <ComplexOptionSelection>
               <OptionName>ECS-TEST PKG1</OptionName>
               <ComplexValue>
                  <SimpleOptionSelection>
                     <OptionName>Production Option</OptionName>
                     <Value>Native Granule</Value>
                  </SimpleOptionSelection>
                  <ComplexOptionSelection>
                     <OptionName>Media Type</OptionName>
                     <ComplexValue>
                        <ComplexOptionSelection>
                           <OptionName>CDROM</OptionName>
                           <ComplexValue>
                              <SimpleOptionSelection>
                                 <OptionName>Compression</OptionName>
                                 <Value>GZip</Value>
                              </SimpleOptionSelection>
                              <SimpleOptionSelection>
                                 <OptionName>DDISTMEDIAFMT</OptionName>
                                 <Value>Rockridge</Value>
                              </SimpleOptionSelection>
                           </ComplexValue>
                        </ComplexOptionSelection>
                     </ComplexValue>
                  </ComplexOptionSelection>
               </ComplexValue>
            </ComplexOptionSelection>
         </OptionSelection>
      </OrderLineItem>
   </CreateOrderRequest>
</OrderEntryService>
```

3.20.2 AddOrderLineItem

This transaction is used to add one or more order line items to an order. Since each order line item uniquely represents a catalog item that is intended to be ordered, one cannot use this transaction to update an already existing order line item. To modify an order line item within an order, such as changing the quantity and options, use the transaction UpdateOrderLineItem.

Code Example 116: Adding two order line items without options.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <AddOrderLineItemRequest>
      <OrderID>3103</OrderID>
      <OrderLineItem>
         <CatalogItemID>758</CatalogItemID>
         <quantityOrdered>3</quantityOrdered>
      </OrderLineItem>
      <OrderLineItem>
         <CatalogItemID>759</CatalogItemID>
         <quantityOrdered>4</quantityOrdered>
      </OrderLineItem>
   </AddOrderLineItemRequest>
</OrderEntryService>
```

Code Example 117: Adding a single order line item with options.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <AddOrderLineItemRequest>
      <OrderID>1194182309</OrderID>
      <OrderLineItem>
         <CatalogItemID>G2079-ORNL</CatalogItemID>
         <quantityOrdered>4</quantityOrdered>
         <OptionSelection>
            <ComplexOptionSelection>
               <OptionName>ORNL Package Options</OptionName>
               <ComplexValue>
                  <ComplexOptionSelection>
                     <OptionName>Media Type</OptionName>
                     <ComplexValue>
                        <ComplexOptionSelection>
                           <OptionName>FTP_Pull</OptionName>
                           <ComplexValue>
                              <SimpleOptionSelection>
                                <OptionName>Compression</OptionName>
                                <Value>TAR</Value>
                              </SimpleOptionSelection>
                           </ComplexValue>
                        </ComplexOptionSelection>
                     </ComplexValue>
                  </ComplexOptionSelection>
               </ComplexValue>
            </ComplexOptionSelection>
```

```
        </OptionSelection>
      </OrderLineItem>
   </AddOrderLineItemRequest>
</OrderEntryService>
```

3.20.3 DeleteOrder

This transaction is used to delete one or more orders from the system. A user can only delete orders that are in the pre-submittal phase, i.e., orders that have not yet been submitted. For post-submittal orders, a registered user can use the CancelOrder operation in the User Account Service to cancel an open order. Guest orders can not be deleted in the post-submittal phase. This transaction will completely delete the entire order, including all associated provider orders. To delete a specific provider order(s), use the DeleteProviderOrder transaction.

Code Example 118: Example of deleting three orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <DeleteOrderRequest>
      <OrderID>1234</OrderID>
      <OrderID>2345</OrderID>
      <OrderID>3456</OrderID>
   </DeleteOrderRequest>
</OrderEntryService>
```

3.20.4 DeleteOrderLineItem

This transaction is used to remove one or more order line items from an order.

Code Example 119: Deleting two order line items from an order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <DeleteOrderLineItemRequest>
      <OrderID>3103</OrderID>
      <CatalogItemID>758</CatalogItemID>
      <CatalogItemID>759</CatalogItemID>
   </DeleteOrderLineItemRequest>
</OrderEntryService>
```

3.20.5 DeleteProviderOrder

This transaction is used to remove a provider order from a larger order context. It will only delete the specified provider order (and thus all of the order line items associated with that provider) from the order. All other non-specified provider orders will stay intact.

Code Example 120: Deleting a specific provider order from an order.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <DeleteProviderOrderRequest>
      <ProviderOrderID>
         <ProviderID>ECHO-TEST</ProviderID>
         <OrderID>3955</OrderID>
      </ProviderOrderID>
   </DeleteProviderOrderRequest>
</OrderEntryService>
```

3.20.6 ListUnsubmittedOrderSummary

This transaction is used to present for a user a summary of orders that they have created, not yet submitted, and matches one of the specified order states passed to it. This transaction only returns the ID and state of each order that meets the criteria. If no unsubmitted order state is passed to the transaction, then all the orders that have been created but not yet submitted for that user will be presented.

Note: This is one of the few transactions in the OrderEntryService that a guest does not have access to. Only a registered user (this does not include providers) can use this transaction.

Code Example 121: Presenting orders in two unsubmitted states: NOT_VALIDATED and VALIDATED.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <ListUnsubmittedOrderSummaryRequest>
      <UnsubmittedOrderState>
         <NOT_VALIDATED/>
      </UnsubmittedOrderState>
      <UnsubmittedOrderState>
         <VALIDATED/>
      </UnsubmittedOrderState>
   </ListUnsubmittedOrderSummaryRequest>
</OrderEntryService>
```

Code Example 122: Presenting orders in all unsubmitted states.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <ListUnsubmittedOrderSummaryRequest/>
</OrderEntryService>
```

3.20.7 PresentCatalogItem

This transaction is used to obtain information about one or more catalog items. The information available for a catalog item includes:

- Type and description of the item,
- Which provider owns it,

- Price (if any),
- Options (required and available) for the item. These options may include shipping and packaging options, or the information necessary for the fulfillment of a service.

Each ordered catalog item creates one order line in a user's order.

Code Example 123: Presenting a catalog item.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentCatalogItemRequest>
      <CatalogItemID>G2076-ORNL</CatalogItemID>
   </PresentCatalogItemRequest>
</OrderEntryService>
```

3.20.8 *PresentOptionDefinitionsForOrder*

This transaction is used to request a list of the options (required and available) that may be selected for a particular order. These options may include shipping and packaging options, and/or payment information. They apply to all the provider orders within the order.

Note: This transaction is not fully implemented for ECHO release 5.5.

3.20.9 *PresentOptionDefinitionsForProviderOrder*

This transaction is used to present the provider-level option selections for a specific provider order. These selections may include shipping and packaging options, and will be applied to all order line items that contain provider-specific options within that provider order. If no option names are declared, all the options are displayed. Use the PresentOrderOptionDefinitionsForProvider transaction to view the possible options one can set for this provider. Use the SetProviderOptionSelectionsForProviderOrder transaction to set the options.

Code Example 124: Presenting option selections for provider order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentProviderOptionSelectionsForProviderOrderRequest>
      <ProviderOrderID>
         <ProviderID>P14381</ProviderID>
         <OrderID>1731053521</OrderID>
      </ProviderOrderID>
      <OptionName>Special Instructions</OptionName>
   </PresentProviderOptionSelectionsForProviderOrderRequest>
</OrderEntryService>
```

3.20.10 *PresentOrder*

This transaction is used to present the complete specification and description of an order that is being built or to see the status of an order that has already been submitted. The user can use the PresentProviderOrder transaction to see just the portion of the order that is specific to a particular provider.

Code Example 125: Presenting three orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentOrderRequest>
      <OrderID>1234</OrderID>
      <OrderID>2345</OrderID>
      <OrderID>3456</OrderID>
   </PresentOrderRequest>
</OrderEntryService>
```

3.20.11 *PresentOrderOptionDefinitionsForProvider*

This transaction is used to request a list of the options (required and available) that may be selected for a particular provider. These options may include shipping and packaging options, and/or payment information. Currently this transaction is not fully implemented.

3.20.12 *PresentProviderOrder*

This transaction is used to present the portion(s) of an order that is fulfilled by one particular provider. All orders are broken into these sub-orders with each sub-order grouped by a provider.

Code Example 126: Presenting two orders from the same provider.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <PresentProviderOrderRequest>
      <ProviderOrderID>
         <ProviderID>ECHO-TEST</ProviderID>
         <OrderID>3202</OrderID>
      </ProviderOrderID>
      <ProviderOrderID>
         <ProviderID>ECHO-TEST</ProviderID>
         <OrderID>3252</OrderID>
      </ProviderOrderID>
   </PresentProviderOrderRequest>
</OrderEntryService>
```

3.20.13 *QuoteOrder*

This transaction is a request for a particular order to be quoted. A quoted order provides a binding price for the order as a whole. A quote request is a non-blocking request, as the binding quote response from each of the necessary providers may take a day or more to receive. No changes may be made to an order while it is in the quoting process. Also, a change to the order subsequent to the receipt of a quote may invalidate the quoted price.

Code Example 127: Quoting on a single order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
```

```
<OrderEntryService>
   <QuoteOrderRequest>
      <OrderID>2959</OrderID>
   </QuoteOrderRequest>
</OrderEntryService>
```

Notes:

The QuoteOrder transaction only works on validated orders. An order must first be validated through the ValidateOrder transaction (see Section 6.1.20) before the QuoteOrder transaction can take place.

Currently, since no providers support quoting an order, the QuoteOrder transaction will not update the price on any catalog item.

3.20.14 SetAuthenticationKey

This transaction is used to set the authentication key for a provider order. The authentication key is a provider-issued password that grants the user access to data.

Code Example 128: Setting the authentication key.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <SetAuthenticationKeyRequest>
      <ProviderOrderID>
         <ProviderID>ECHO-TEST</ProviderID>
         <OrderID>3202</OrderID>
      </ProviderOrderID>
      <AuthenticationKey>testAuthenticationKey</AuthenticationKey>
   </SetAuthenticationKeyRequest>
</OrderEntryService>
```

3.20.15 SetOptionSelectionsForOrder

This transaction is used to set a list of options that have been defined for a particular order. These options may include shipping and packaging options, and/or payment information and is applied to all the provider orders within that order. Use PresentOptionDefinitionsForOrder transaction to view the possible options one can set for this order.

Note: This transaction is not fully implemented for ECHO release 5.5.

3.20.16 SetOptionSelectionsForProviderOrder

This transaction is used to set the provider-level options for a specific provider order. These options may include shipping and packaging options, and will be applied all order line items that contain provider-specific options within that provider order. Use the PresentOrderOptionDefinitionsForProvider transaction to view the possible options one can set for this provider.

Code Example 129: Set provider option selections for a provider order.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
```

```
<OrderEntryService>
   <SetProviderOptionSelectionsForProviderOrderRequest>
      <ProviderOrderID>
         <ProviderID>1000</ProviderID>
         <OrderID>3955</OrderID>
      </ProviderOrderID>
      <OptionSelection>
         <ComplexOptionSelection>
            <OptionName>ECS-TEST PKG1</OptionName>
            <ComplexValue>
               <SimpleOptionSelection>
                  <OptionName>Production Option</OptionName>
                  <Value>Native Granule</Value>
               </SimpleOptionSelection>
               <ComplexOptionSelection>
                  <OptionName>Media Type</OptionName>
                  <ComplexValue>
                     <ComplexOptionSelection>
                        <OptionName>CDROM</OptionName>
                        <ComplexValue>
                           <SimpleOptionSelection>
                              <OptionName>Compression</OptionName>
                              <Value>GZip</Value>
                           </SimpleOptionSelection>
                           <SimpleOptionSelection>
                              <OptionName>DDISTMEDIAFMT</OptionName>
                              <Value>Rockridge</Value>
                           </SimpleOptionSelection>
                        </ComplexValue>
                     </ComplexOptionSelection>
                  </ComplexValue>
               </ComplexOptionSelection>
            </ComplexValue>
         </ComplexOptionSelection>
      </OptionSelection>
   </SetProviderOptionSelectionsForProviderOrderRequest>
</OrderEntryService>
```

3.20.17 *SetUserInformationForOrder*

Every order must have its own user-specific information attached to it so that a data provider can process the order. Contact Address (which includes a user's name, address, phone number, and email address) is required. Billing and shipping address are both optional. Each order is independent of other orders, so user information must be set for every order created.

Again, the system stresses flexibility. User information is not required until you either validate the order or actually submit the order. At any time before submission, the user has the option of providing some or none of the user information needed for the order. The user can also update and delete order line items from a specific order either before or after setting the user information. However, eventually all of this information will be necessary before the provider processes an order. If the required information is not present at the time of order validation or submission, an error WILL be sent to the user. Thus before submitting or validating an order, make sure that the required user information for that particular order is completely filled out. To do this, one must use the SetUserInformationForOrder transaction within the OrderEntryService.

Code Example 130: Set User Information for order

```
<?xml version="1.0"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <SetUserInformationForOrderRequest>
      <OrderID>3955</OrderID>
      <ShippingAddress>
         <shipToBusinessName>Blueprint</shipToBusinessName>
         <ContactName>
            <FirstName>Michael</FirstName>
            <LastName>Hudson</LastName>
         </ContactName>
         <specialInstruction>Please leave the package at the front door, not in the mailbox.</specialInstruction>
         <EmailString>mikeh@userguide.com</EmailString>
         <AddressInformation>
            <AddressID>Shipping</AddressID>
             <USFormat>TRUE</USFormat>
            <Street1>7600 Kingsbury Road</Street1>
            <Street2/>
            <City>Alexandria</City>
            <State>VA</State>
            <Zip>22315</Zip>
            <Country>USA</Country>
         </AddressInformation>
      </ShippingAddress>
      <BillingAddress>
         <ContactName>
        <FirstName>Michael</FirstName>
        <LastName>Hudson</LastName>
         </ContactName>
         <EmailString>billg@microsoft.com</EmailString>
         <AddressInformation>
            <AddressID>Billing</AddressID>
            <USFormat>TRUE</USFormat>
            <Street1>7600 Kingsbury Road</Street1>
            <Street2/>
            <City>Alexandria</City>
            <State>VA</State>
            <Zip>22315</Zip>
            <Country>USA</Country>
         </AddressInformation>
      </BillingAddress>
      <ContactAddress>
         <ContactName>
        <FirstName>Michael</FirstName>
        <LastName>Hudson</LastName>
         </ContactName>
         <PhoneString>703-921-9393</PhoneString>
         <EmailString>mhudson@blueprinttech.com</EmailString>
         <AddressInformation>
            <AddressID>Contact</AddressID>
            <USFormat>TRUE</USFormat>
            <Street1>7600 Kingsbury Road</Street1>
            <City>Alexandria</City>
            <State>VA</State>
            <Zip>22315</Zip>
            <Country>USA</Country>
         </AddressInformation>
      </ContactAddress>
   </SetUserInformationForOrderRequest>
</OrderEntryService>
```

3.20.18 SubmitOrder

At this point, the order should be complete and valid, with all appropriate options filled out and the associated user information completed. The order may also have been quoted at this point as well. If all the previous actions have executed successfully, and no other changes have been made to the order, then the order is ready to be submitted to the system. To do this, use the SubmitOrder transaction in the OrderEntryService.

Code Example 131: SubmitOrder transaction

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <SubmitOrderRequest>
      <OrderID>1002</OrderID>
      <NotificationLevel>
         <VERBOSE/>
      </NotificationLevel>
   </SubmitOrderRequest>
</OrderEntryService>
```

The NotificationLevel is the place where user specifies his/her preference for receiving the updated status of a particular order. The choices are

- **VERBOSE:** the system will email all provider order state changes and status message updates to the user;
- **DETAIL:** the system will email the user when only provider order state changes are made;
- **INFO:** the system will email the user when the provider order is closed or cancelled;
- **CRITICAL:** the system will email the user when the order is failed in submitting or get rejected by the provider;
- **NONE:** the system will not send the user email. The user will have to come back to ECHO and use transactions like 'PresentOrder' to find out about the status of the order.

For any choice, the user can always come back to ECHO and check the status of the order through the 'PresentOrder' transaction. If the user does not specify any value of this data type, for a registered user's order, the user's preference value will be used; for guests' orders, or if no preference value is set for a registered user, the default value is VERBOSE.

Once the order is transmitted to the system using the Submit transaction, each provider order within the greater order is sent to their appropriate providers. It is then up to each provider whether they will accept the provider order and/or how they process the provider order. From this point on, the entire order is out of the hands of the ECHO. However, one can track the status of the order using the PresentOrder transaction in the OrderEntryService. At any point before the order is actually shipped, the user may request cancellation of their order through the CancelOrder transaction in the UserAccountService. The user can then use the PresentOrder transaction again to track whether the order was successfully cancelled.

When ECHO sends a request to a provider to either quote, submit, or cancel a particular order, it is necessary that both ECHO and that provider have the same ID for that order. Within ECHO, the specific provider order is denoted by the ProviderOrderID (which is just the combination of the provider name and the order ID). However, the provider may also have its own ID system for tracking a particular provider order. As a solution, the provider order in ECHO also contains a provider-tracking ID. When the request is sent to the provider from ECHO, the provider has the option to either accept the order ID that ECHO has already given that provider order, or provide ECHO with its own unique ID. If the provider does pass back its own unique ID as the provider-tracking ID, then ECHO will also use that ID in addition to the normal ECHO ProviderOrderID to refer to this provider order. If the provider does not specify a tracking ID, then it is assumed that the order ID that ECHO currently uses will suffice, and the provider order will be referenced using just the ECHO ProviderOrderID instead.

3.20.19 UpdateOrderLineItem

This transaction is used to modify one or more order line items, including the quantity and the options associated and available for that order line item.

Code Example 132: Updating quantity of an order line item.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <UpdateOrderLineItemRequest>
      <OrderID>3103</OrderID>
      <OrderLineItem>
         <CatalogItemID>731</CatalogItemID>
         <quantityOrdered>4</quantityOrdered>
      </OrderLineItem>
   </UpdateOrderLineItemRequest>
</OrderEntryService>
```

3.20.20 ValidateOrder

One characteristic of the ordering system with ECHO is flexibility. One can create, update, and delete orders at any time as well as add, change, and delete order options at any time. There are no specific restrictions on any of these actions at any time until one decides that the order creation is over and they want to submit the order to the provider. At this time, an order must first be validated. This is to ensure that the order is complete and correct – that all necessary and required options are filled out, and the required user-associated information for that order is appropriately completed. If one attempts to try to submit a non-validated order, an error message will be returned to the user saying as much. Currently, only the first validation error encountered is returned when trying to validate, so revalidation is necessary once the reported error is fixed to fully validate the order. Also, if any aspect of a validated order is changed before it is submitted, then the order becomes non-validated again. To validate an order, use the ValidateOrder transaction in the OrderEntryService (see Code Example 133).

After validation, it may be possible (depending on whether the provider supports this transaction) to get the exact price of each provider order before you actually submit the order. To do this, run the QuoteOrder transaction in the OrderEntryService (see Section 3.18).

Notes:

1.The QuoteOrder transaction only works on validated orders. An order must first be validated through the ValidateOrder transaction before the QuoteOrder transaction can take place.

2.Currently, since no providers support quoting an order, the QuoteOrder transaction will not update the price on any catalog item.

Keep in mind that because many of the validation rules can be provider specific and change over time, running the ValidateOrder transaction may often return a validation error. However, the error messages should be complete enough to help you determine what aspects of the order need to be filled out or corrected.

It is important to understand that you may have to run the order through this transaction often, each time obtaining an error message and correcting the problem appropriately. In general, the error messages cover whether required options for a catalog item, provider-order, or order are not filled out, or whether a piece of user information was not completed. Also, error messages can occur if the order specified does not exist, doesn't belong to the current user, or the order itself has already been submitted or quoted.

Code Example 133: ValidateOrder transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <ValidateOrderRequest>
      <OrderID>4104</OrderID>
   </ValidateOrderRequest>
</OrderEntryService>
```

3.21 Notifications

ECHO users do not interact with the providers directly. However, actions by users –submitting, quoting, and canceling an order – require that the providers either accept or reject the action. If such actions are initiated, ECHO will communicate with the providers for the decision. Note that users normally receive a REQUEST SUCCEEDED BooleanResultType in the response message. This only means that ECHO successfully received the request from the user. The acceptance or rejection of the request is the responsibility of the provider. As the response from each associated provider may take a day or more to receive, the status of each provider order and the order at large is not known automatically. Users can track the providers' decisions by looking at the timestamped StatusMessage in the PresentOrder response.

There are three specific cases where user initiated transactions will offer further notification possibilities.

QuoteOrder. If a specific provider accepts the quote request, the supplied quote information will be shown in the ProviderQuote under the ProviderOrder belonging to that provider in the PresentOrder response.

SubmitOrder. The user can request to be notified of status changes by email. However, to see the order information (such as price, expected shipping date, etc.) use the PresentOrder transaction.

CancelOrder. Users can check the status of the request using the PresentOrder transaction. If a user has requested to be notified of status changes by email at the level VERBOSE, DETAIL, or INFO, the user will be notified once the order is cancelled.

3.22 Order Entry Service Error Messages

Table 3-13: Order Entry Service Error Messages

Transaction	Error Message	Description
AddOrderLineItem	[<OptionName>] is not a valid property for this item	An invalid option is associated with an item.
	gov.nasa.echo.business.ejb.order. OrderException (Cannot add an item with quantity less than or equal to 0).	The user submitted an item for an order with a negative or zero quantity.
	The catalog item(s) <CatalogItemIDs> has (have) been deleted by the provider. Please do not add the item(s) to the order.	The user tried to add a catalog item that has been deleted by the provider.
	The catalog item(s) <CatalogItemIDs> is (are) not permitted to order. Please do not add the item(s) to the order.	The user tried to add an order-restricted item to an order.
	There is no property <ChildOption> defined for the type: <ParentOption>	Error returned when an invalid option structure is associated with an item.
	Unable to get Catalog Item [<CatalogItemID>] because Invalid CatalogItem id: <CatalogItemID>.	The CatalogItemID submitted is not valid.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
CreateOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
DeleteOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
DeleteOrderLineItem	Unable to get Catalog Item [<CatalogItemID>] because Invalid CatalogItem id: <CatalogItemID>.	The CatalogItemID submitted is not valid.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
DeleteProviderOrder	Find Provider: (<ProviderID>) for ProviderOrder resulted in error.	The user submitted a ProviderID that does not exist.
	There is no property <ChildOption> defined for the type: <ParentOption>	The ProviderID submitted does not exist..
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
PresentCatalogItem	The catalog item(s) <CatalogItemIDs> has (have) been deleted by the provider.	A requested item has been deleted by the provider.

Transaction	Error Message	Description
	The catalog item(s) <CatalogItemIDs> is(are) not permitted to order.	An item ordered is restricted to the user.
	Unable to find catalog item [<catalogItemIDs>]	The user tried to request information for an item that does not exist.
PresentOptionDefinitionsForOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unable to locate order [<OrderID>].	The OrderID submitted is not valid for the provider.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
PresentOptionDefinitionsFor-ProviderOrder	Find Provider: (<ProviderID>) for ProviderOrder resulted in error.	The user submitted a ProviderID that does not exist.
	There is no property <ChildOption> defined for the type: <ParentOption>	The ProviderID submitted does not exist..
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
PresentOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
PresentProviderOrder	Find Provider: (<ProviderID>) for ProviderOrder resulted in error.	The user submitted a ProviderID that does not exist.
	There is no property <ChildOption> defined for the type: <ParentOption>	The ProviderID submitted does not exist..
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
QuoteOrder	Order must be validated before quoting.	The user submitted a request for an order quote before validating the order.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
SetOptionSelectionsForOrder	“[<OptionName>] is not a valid property for this item	The user tried to access an option that is not valid for an item.
	“[<OptionName>] is not a valid property for this item	The user submitted an invalid option structure.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.

Transaction	Error Message	Description
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
SetOptionSelectionsForProvider-Order	Find Provider: (<ProviderID>) for ProviderOrder resulted in error.	The user submitted a ProviderID that does not exist.
	"[<OptionName>] is not a valid property for this item	The user tried to access an option that is not valid for an item.
	There is no property <ChildOption> defined for the type: <ParentOption>	The user submitted an invalid option structure.
	There is no property <ChildOption> defined for the type: <ParentOption>	The ProviderID submitted does not exist..
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
SetUserInformation ForOrder	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
SubmitOrder	Order must be validated or quoted before submitting.	The user submitted an order before validating the order.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.
UpdateOrderLineItem	"[<OptionName>] is not a valid property for this item	An invalid option is associated with an item
	gov.nasa.echo.business.ejb.order. OrderException (Cannot add an item with quantity less than or equal to 0).	The user submitted an item for an order with a negative or zero quantity.
	The catalog item(s) <CatalogItemIDs> has (have) been deleted by the provider. Please do not add the item(s) to the order.	The user tried to add a catalog item that has been deleted by the provider.
	The catalog item(s) <CatalogItemIDs> is (are) not permitted to order. Please do not add the item(s) to the order.	The user tried to add an order-restricted item to an order.
	There is no property <ChildOption> defined for the type: <ParentOption>	Error returned when an invalid option structure is associated with an item.
	Unable to get Catalog Item [<CatalogItemID>] because Invalid CatalogItem id: <CatalogItemID>.	The CatalogItemID submitted is not valid.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.

Transaction	Error Message	Description
ValidateOrder	No shipping address specified.	The user submitted an order without the required shipping address information.
	Order options not defined for item [<CatalogItemID>].	The user submitted an order without the required options.
	Unable to locate order [<OrderID>].	The OrderID submitted does not exist.
	Unauthorized access to order.	The user tried to access an order that does not belong to that user.

3.23 Order State Conversions

Order states are based on the provider order state. An order consists of zero or many provider orders. An order state is calculated from provider order state(s). Table 3-14 shows the states for orders made up of zero or one provider order, or multiple provider orders having the same provider order state.

Table 3-14: Order State Conversion: Zero or One Provider Order, or Multiple Provider Orders having the same provider order state.

# of Provider Orders	Provider Order State	Order State
0	---	NOT_VALIDATED
1	---	Same as provider order state
	QUOTE_FAILED	QUOTED_WITH_EXCEPTIONS
	QUOTE_REJECTED	
	SUBMIT_FAILED	SUBMITTED_WITH_EXCEPTIONS
	SUBMIT_REJECTED	

Figure 3-8 shows how to determine the state of an order made up of more than one provider order.

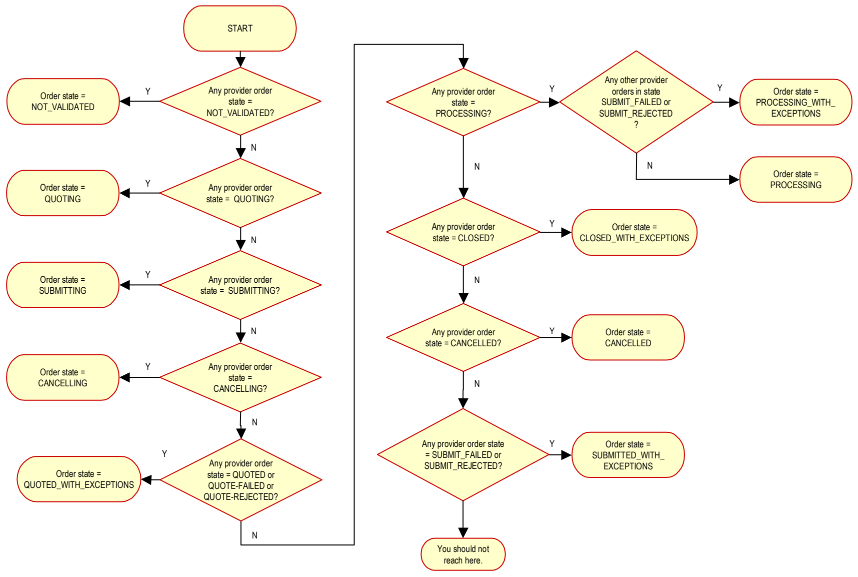

START
Any provider order state = NOT_VALIDATED?
Y
Order state = NOT_VALIDATED
N
Any provider order state = QUOTING?
Y
Order state = QUOTING
N
Any provider order state = SUBMITTING?
Y
Order state = SUBMITTING
N
Any provider order state = CANCELLING?
Y
Order state = CANCELLING
N
Any provider order state = QUOTED or QUOTE-FAILED or QUOTE-REJECTED?
Y
Order state = QUOTED_WITH_EXCEPTIONS
N
Any provider order state = PROCESSING?
Y
Any other provider orders in state SUBMIT_FAILED or SUBMIT_REJECTED ?
Y
Order state = PROCESSING_WITH_EXCEPTIONS
N
Order state = PROCESSING
N
Any provider order state = CLOSED?
Y
Order state = CLOSED_WITH_EXCEPTIONS
N
Any provider order state = CANCELLED?
Y
Order state = CANCELLED
N
Any provider order state = SUBMIT_FAILED or SUBMIT_REJECTED?
Y
Order state = SUBMITTED_WITH_EXCEPTIONS
You should not reach here.

Figure 3-8 Order state conversion: more than one provider order.

3.24 Caveats in the Order Entry Service

3.24.1 Restriction on order items

Those with permission may only order some catalog items. The provider sets the restriction and permissions on catalog item ordering. Restrictions can apply to individual users, groups of users, or all users. If a catalog item is not available to a particular user and that user executes the PresentCatalogItem, CreateOrder, or AddOrderLineItem transaction including this catalog item, a message will show to the user that it is not an orderable item. If a provider change the restriction/permission on an once orderable item to not orderable after the order has been created, the user who has the item in his/her order will be asked to remove this item from the order when the user UpdateOrderLineItem, ValidateOrder, QuoteOrder, or SubmitOrder.

3.24.2 Deleted Items

Providers can delete their data items. ECHO may have records for these items, but they can no longer be ordered. If a user tries to PresentCatalogItem, CreateOrder, or AddOrderLineItem including the deleted catalog item, an error message will be returned, indicating that it is deleted item. If a provider deletes an item after the order has been created, the user who has the item in his/her order will be asked to remove this item from the order when the user executes the UpdateOrderLineItem, ValidateOrder, QuoteOrder, or SubmitOrder transaction.

3.25 Subscription Service

3.26 Transactions

The Subscription Service allows ECHO users to subscribe to metadata from different providers. When providers update the metadata repository, ECHO delivers the updated metadata to each subscriber. Although this capability is useful for end users, it can be extremely valuable for client developers, because it provides an extension point from ECHO. A subscription can be created to deliver new metadata to a client infrastructure that performs post-processing and data massaging in order to convert the data into a more usable form.

The three keys in creating a subscription are:

1. **What dataset are you interested in?** To determine what providers exist in the system, you can use the ListAllProviders transaction in the Provider Profile Service. Additionally, the Catalog Service discovery mechanism retrieves the datasets currently stored in the metadata repository. For more information, see the documentation for the respective services.
2. **What type of metadata from the dataset are you interested in?** ECHO allows a user to subscribe to collection level metadata, granule metadata, or both. Collection metadata will rarely change whereas granule metadata will frequently be updated and/or created. Most users choose to subscribe to granule metadata for this reason. In addition, from time to time, granule metadata may be deleted by the provider. If this happens, the users will receive a notification email from ECHO about the granule deletion.
3. **Where should ECHO send the metadata when it is updated?** ECHO supports two methods of subscription delivery: FTP and E-Mail. Due to security restrictions, we require all FTP deliveries to be transferred via passive-mode. If delivery cannot be made (because the FTP server is full or the mail server rejected the large file attachment), ECHO sends a notification e-mail to the subscriber. Consult your local administrator for more information on passive-mode transfers and file size limitations.

The Subscription Service supports a full set of transactions that allow the subscribers to easily manage their subscriptions. The following sections detail the transactions in metadata subscription service.

3.26.1 CreateSubscription

To receive ECHO metadata updates, the first step is creating a metadata subscription by sending CreateSubscriptionRequest message to ECHO.

Code Example 134: Metadata subscription

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <CreateSubscriptionRequest>
      <MetadataSubscription>
         <SubscriptionName>myFirstSubscription</SubscriptionName>   ← the name of the subscription
         <providerName> ORNL-DAAC</providerName>           ← the provider we are interested in
         <datasetName>MODIS Geolocation V001</datasetName>      ← the dataset we are interested in

         <MetadataFilterInfo>
                   <spatialCondition>GLOBAL</spatialCondition>      ← indicates that no spatial restriction should be
applied to the metadata
         </MetadataFilterInfo>

         <MetadataActionInfo>
            <PresentationDescription>
               <DTDType><ECHO/></DTDType>                             ← the type of format ECHO should use
            </PresentationDescription>

            <CompressionType><GZIP/></CompressionType>           ← tells ECHO how to compress the data  before sending it
to the client

            <SubscriptionUpdateType><BOTH/></SubscriptionUpdateType>     ← the type of data we are interested in (see Key
#2)
         </MetadataActionInfo>

         <DeliveryInfo>
            <DeliveryType>   <FTPPUSH/></DeliveryType>     ← the delivery mechanism
            <deliveryAddress>anonymous@ftp.gst.com</deliveryAddress>    ← always of the form user@host.com
            <password>foo</password>    ← not required if delivery is EMAIL
          <deliveryFolder>/pub/incoming/echo</deliveryFolder>    ← not required if delivery is EMAIL
            <limitSize>4.0</limitSize>      ← allows you to restrict the size of a subscription.  If the size of the file
to deliver exceeds this value, no delivery is attempted.
         </DeliveryInfo>

         <StopDate>          ← the expiration date
            <Date>
               <Month><MonthName><SEPTEMBER/></MonthName></Month>
               <Day>30</Day>
               <Year>2003</Year>
            </Date>
         </StopDate>
      </MetadataSubscription>
   </CreateSubscriptionRequest>
</SubscriptionService>
```

3.26.1.1 SubscriptionName

A unique name given to each subscription. No two subscriptions belonging to the same user may have the same name.

3.26.1.2 ProviderName

The UserID of the provider who's data we are interested in.

3.26.1.3 DataSetName

The specific dataset we are interested in. This can be discovered through the Catalog Service.

3.26.1.4 MetaDataFilterInfo

Contains the spatial constraint to use for the subscription. This allows a user to narrow the data they are interested in to a specific geographic location.

3.26.1.5 MetaDataActionInfo

Contains the information related to processing of the subscription. The PresentationDescription allows a user to specify a DTD format to which the XML sent to the client must conform. Users can also put their interested attribute names in the TupleType list. For example, if a user is only interested in the "platform" metadata, they can use the TupleType parameter to indicate this. In the above example, the user wants to receive data in "ECHO" format, with all available attributes to be presented. The TupleType validation is based on collection DTDs if the subscription update type is ALL_COLLECTIONS or COLLECTIONS_ONLY. The TupleType validation is based on granule DTDs if the subscription update type is GRANULES_ONLY or BOTH.

3.26.1.6 CompressionType

Allows users to specify any compression they desire on the delivered metadata file. It should be noted that the subscription files delivered from ECHO are XML formatted and hence highly compressible. It is highly recommended that users use a CompressionType of GZIP to conserve space and bandwidth.

3.26.1.7 SubscriptionUpdateType

Allows the user to subscribe to four types of metadata:

1. All Collection Metadata: The subscribers will receive every updated collection metadata. This type refers to ALL_COLLECTIONS for SubscriptionUpdateType.
2. A Specific Collection's Collection Metadata: The subscribers will only receive updated collection metadata of a subscriber-specified collection. This type refers to COLLECTIONS_ONLY for SubscriptionUpdateType.
3. A Specific Collection's Granule Metadata: The subscriber will receive every updated granule metadata in a subscriber-specified collection. This type refers to GRANULES_ONLY for SubscriptionUpdateType.
4. A Specific Collection's Granule and Collection Metadata: The subscriber will receive both the updated collection metadata and every updated granule metadata in a subscriber-specified collection. This type refers to BOTH for SubscriptionUpdateType.

3.26.1.8 DeliveryInfo

The Subscription Service allows the user to specify how metadata should be delivered to them. Currently, ECHO supports Email and FTP as delivery mechanisms. If Email is chosen, the deliveryAddress should be of the format "username@host.com". If FTPPUSH is chosen, the deliveryAddress should also be of the format "username@ftp.host.com". For FTP deliveries, the deliveryFolder and Password fields are required. In the above example, the user wants the metadata to be delivered via FTP push to the server ftp.gst.com, with the user "anonymous" with password "foo". The metadata sent from ECHO will be placed in the delivery folder "/pub/incoming/echo".

The Subscription Service has a governor that prevents delivery of metadata above a certain threshold. The limitSize is the max size (in megabytes) of the delivery file that the user will accept. The size of the data file is calculated prior to delivery. If the size of the data file exceeds the user specified limitSize, the system does not deliver the file. In addition, the user's email address in the user's profile is used in order that information about their subscription can be sent to them (such as notices when it expires, or if the file to be sent exceeds the set limitation on size). Also, after the data is uploaded to the registered ftp site (if FTP Push is selected), notice is also sent to the user at the email address.

3.26.1.9 StopDate

Each subscription has an expiration date that is specified in the StopDate. This field allows a user to specify a date after which no metadata should be delivered. In this example, the subscription will expire on September 30, 2003.

3.26.2 DeleteSubscription

Code Example 135: Deleting a subscription

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <DeleteSubscriptionRequest>
      <SubscriptionName>myFirstSubscription</SubscriptionName>
   </DeleteSubscriptionRequest>
</SubscriptionService>
```

3.26.3 ListSubscriptionNames

To enable management of the multiple subscriptions, the user can use ListSubscriptionNames transaction to list all the active subscription names.

Code Example 136: Listing active subscriptions

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <ListSubscriptionNamesRequest/>
</SubscriptionService>
```

3.26.4 PauseSubscription

This transaction will pause an active metadata subscription. No metadata updates will be received for a paused subscription. A paused subscription can be resumed using ResumeSubscription transaction.

Code Example 137: Pausing an active subscription

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <PauseSubscriptionRequest>
      <SubscriptionName>myFirstSubscription</SubscriptionName>
   </PauseSubscriptionRequest>
</SubscriptionService>
```

3.26.5 PresentSubscription

Presenting a subscription

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <PresentSubscriptionRequest>
      <SubscriptionName>myFirstSubscription</SubscriptionName>
   </PresentSubscriptionRequest>
</SubscriptionService>
```

3.26.6 ResumeSubscription

Code Example 138: Resuming a paused subscription

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE SubscriptionService PUBLIC "-//ECHO SubscriptionService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/SubscriptionService.dtd">
<SubscriptionService>
   <ResumeSubscriptionRequest>
      <SubscriptionName>myFirstSubscription</SubscriptionName>
   </ResumeSubscriptionRequest>
</SubscriptionService>
```

3.26.7 UpdateSubscription

This transaction updates the information for an existing Metadata Subscription. Active, paused, and expired subscriptions can all be updated.

3.26.8 ListExpiredSubscriptionNames

This transaction generates a list of the names of all of the user’s expired metadata subscriptions. The response includes an indication of whether the action succeeded, as well as the names of all expired metadata subscriptions associated with the user.

3.26.9 ListPausedSubscriptionNames

This transaction generates a list of the names of all of the user’s paused metadata subscriptions. The response includes an indication of whether the action succeeded, as well as the names of all paused metadata subscriptions associated with the user.

3.26.10 RenewSubscription

A subscription expires on its stop date, which is identified in the CreateSubscription request. After this date, use this transaction to renew an expired metadata subscription, causing it to become active again. The request message identifies the subscription name and the new stop date. The response indicates whether the action succeeded.

3.27 Subscription Service and Restricted Data

Prior to delivering metadata to subscribers, the Subscription Service honors the visibility restrictions as defined by the access control lists in the Data Management Service. If a granule or collection is updated, but restrictions prevent viewing, the granule or collection will not be delivered to the end user. Moreover, no indication is given that the granule or collection was updated but not delivered to the user because of the restriction.

A current limitation of the Subscription Service is that previously updated restricted metadata is not delivered to the user when the data becomes visible. This “feature” is witnessed under the following condition:

1. (As a user) Create a subscription to collection MOD01
2. (As a provider) Create a Rolling Temporal Restriction against MOD01 during Sept 1 2002 – Sept 31 2002

3. (As a provider) Update the MOD01 collection on Sept 15 2002

Under the above condition, the user will not receive the changes sent on Sept 15, 2002. Furthermore, the user will not receive the changes even after the restriction expires on Sept 31, 2002. Updates to MOD01 that occur post-Sept 31, 2002 will be sent to the user. This limitation is being investigated and may be overcome in the near future.

3.28 Subscription Service Error Messages

Three types of errors can occur when sending a valid request to the Subscription Service:

1. Incorrect subscription status transition: For example, if a user sends a ResumeSubscription request to resume an active subscription, this transaction cannot be granted. ECHO first finds out that the subscription is already in active state, and then includes an error message "The metadatasubscription myFirstSubscription has status active. Cannot perform ResumeMetadata Transaction. Its status must be paused to be resumed" in the response message.
2. Incorrect subscription name: If the user gives an incorrect subscription name – one that either does not belong to this user or does not exist in the ECHO datastore – ECHO generates an error "The user John has no subscription named myFirstSubscription" in the response message.
3. Required fields are empty: In some request messages, even valid XML documents, a required field is left empty. For example, in the CreateSubscription request, the subscription name may be left blank. In this case, ECHO generates an error "SubscriptionName is not valid" in the response message.

Table 3-15 gives the possible error messages associated with each transaction within the Subscription Service.

Table 3-15 Subscription service error messages.

Transaction	Error Message	Description
CreateSubscription	<providerName> is an existing ECHO provider. Please choose a provider from the list of ECHO providers:: <providerName1>,<providerName2>…	This error is returned when the provider name requested does not exist. There is a typo in the message. The correct error message should be "<providerName> is not an existing ECHO provider. Please choose a provider from the list of ECHO providers:: <providerName1>,<providerName2>…"
	<datasetName> is not available from the <providerName> provider	This error is returned when the requested dataset of the provider indicated does not exist.
	DeliveryAddress is invalid. Please check its format. The expected format for the DeliveryAddress is userid@domain.com	This error is returned when delivery email address is invalid meaning the delivery email address does not conform to the format of userid@domain.com.
	Password cannot be null. You need to provide a password for a FTPPUSH delivery.	This error is returned when FTP push was chosen as delivery method but the password for login to the destination is not provided.
	DeliveryFolder cannot be null. You need to provide a DeliveryFolder for a FTPPUSH delivery.	This error is returned when FTP push was chosen as delivery method but the delivery folder at the destination is not provided.
	DeliveryLimit Size is a float measured in Megabytes. Please check its format.	This error is returned when size limitation on deliverable is not expressed as a number.

Transaction	Error Message	Description
	StopDate cannot be before today.	This error is returned when the date for stop subscription indicated is earlier then today's date.
	SubscriptionName: <SubscriptionName> already exists for the user <UserName>.	This error is returned when submit a subscription using a subscription name that already exists for the user.
UpdateSubscription PresentSubscription DeleteSubscription PauseSubscription ResumeSubscription	The user <UserName> has no subscription named <SubscriptionName>.	This error is returned when a user trying to access a subscription that does not exist for the user. All the error messages listed for CreateSubscription except the last one listed apply.
CreateSubscription PresentSubscription DeleteSubscription ListSubscriptionNames PauseSubscription ResumeSubscription	[SessionManager] The service {ProviderOrderManagementService} that you attempted to access either does not exist or you do not have the appropriate access level to use it.	This error is returned when trying to access this transaction by a user who does not have the privilege for this transaction.
PauseSubscription ResumeSubscription	The metadatasubscription -783369493 has status P. Cannot perform PauseMetadata Transaction. Its status must be active to be paused.	This error is returned when a user trying to pause a subscription that is already been paused.

4 PROVIDER INTERFACE

4.1 Provider Registration

Potential Providers may contact the ECHO administrators via the SubmitProviderApplication transaction contained within the RegistrationService. The potential provider should specify an OrganizationName, ProviderContact (described in section 2.2.1), a description of holdings (optional) and services (optional) and any additional information they feel ECHO should know about. Currently, the creation of a provider account is a manual process and is not automated.

4.2 Transaction – SubmitProviderApplication

A potential provider can register with ECHO by submitting a provider application in the Registration Service. The application information includes the provider's organization name, at least one instance of provider contact information and an p description of the provider's data holdings, some description of the provider's service or some other additional information.

Code Example 139: Provider registration application.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE RegistrationService PUBLIC "-//ECHO RegistrationService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/RegistrationService.dtd">
<RegistrationService>
   <SubmitProviderApplicationRequest>
      <ProviderApplication>
         <OrganizationName>GST</OrganizationName>
         <ProviderContact>
            <ContactRole>Manager</ContactRole>
            <ContactFirstName>Echo</ContactFirstName>
           <ContactLastName>Provider</ContactLastName>
        </ProviderContact>
      </ProviderApplication>
   </SubmitProviderApplicationRequest>
</RegistrationService>
```

When the application is submitted, ECHO will respond with an XML message that indicates whether the application succeeded and assigns a temporary tracking id that the potential provider can use to communicate with the ECHO Operations team.

However, there is still additional configuration and testing to complete before this ECHO provider is ready to fully participate. Contact the ECHO operations team for details.

4.3 Provider Account Service

Provider accounts exist in the system and are similar to user accounts. Provider accounts have user names, passwords, address information, and contact information. The primary difference between a Provider account and a normal user account is that a Provider account contains contact entities and a Provider account can control options for their catalog items.

4.4 Transactions

4.4.1 AddContact

The key difference between a user account and a provider account is the ability for a provider account to specify Contact entities. Contact entities are additional pieces of information that a Provider can associate with itself. For example, a Provider might have a Billing Department and a Customer Support Department. These two departments are good candidates for a Contact entity. Each Contact entity has a first and last name, address information, and email address. Thus, a Provider that has geographically separate offices may capture and present that information to ECHO client users.

This transaction enables one or more provider contacts to be added for this ECHO provider. A contact is an individual who has a named role within the organization. It has role name, first name, last name, address information, phone information and email. Each contact is distinct by role name, which identifies this contact's role, like "shipping manager", "order manager".

If a contact named "order manager" is created, emails that are sent to users regarding orders will be "CC'd" to this contact's email address.

Code Example 140: Add provider contact.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <AddContactRequest>
      <ProviderContact>
         <ContactRole>Billing Contact</ContactRole>
         <ContactFirstName>Bill</ContactFirstName>
         <ContactLastName>Manager</ContactLastName>
         <AddressInformation>
            <AddressID>Office</AddressID>
            <USFormat>TRUE</USFormat>
            <Street1>9111 Edmonston Rd</Street1>
            <City>Greenbelt</City>
            <State>MD</State>
            <Zip>20770</Zip>
            <Country>USA</Country>
         </AddressInformation>
         <PhoneInformation>
            <PhoneName>Office</PhoneName>
            <CountryCode>1</CountryCode>
            <AreaCode>301</AreaCode>
            <ExchangeCode>313</ExchangeCode>
            <Phone>0164</Phone>
         </PhoneInformation>
         <EMailAddress>bill@host.com</EMailAddress>
      </ProviderContact>
   </AddContactRequest>
</ProviderAccountService>
```

4.4.2 DeleteContact

This transaction enables provider contacts to be deleted. Contact roles have to be specified in order to delete.

Code Example 141: Delete provider contacts.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
    <DeleteContactRequest>
```

```
      <ContactRole>shipping contact</ContactRole>
    </DeleteContactRequest>
</ProviderAccountService>
```

4.4.3 PresentContacts

This transaction enables provider contacts information to be presented. If contact roles are specified, then only the specified contacts are presented. The contact information includes contact role, first name, last name, address information, phone information and email.

Code Example 142: Present contacts (all).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentContactsRequest/>
</ProviderAccountService>
```

Code Example 143: Present contacts (specific).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentContactsRequest>
      <ContactRole>billing contact</ContactRole>
   </PresentContactsRequest>
</ProviderAccountService>
```

4.4.4 PresentPolicyDefinitions

This transaction enables the provider policy definitions to be presented. Normally, the policy definitions are defined in the ECHO system, and they can be modified based on the needs of all ECHO providers.

Code Example 144: Present policy definitions.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentPolicyDefinitionsRequest/>
</ProviderAccountService>
```

4.4.5 PresentPolicySelections

This transaction enables the provider policy selections to be presented. The provider may specify the policy name. If the policy name is not specified, the complete list of current policy selections will be presented.

Code Example 145: Present policy selections.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentPolicySelectionsRequest/>
</ProviderAccountService>
```

4.4.6 Present Provider Info

This transaction enables the provider information and provider contacts information to be presented. Currently, the provider information just has provider's organization information.

Code Example 146: Present provider information.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <PresentProviderInfoRequest/>
</ProviderAccountService>
```

4.4.7 SetPolicySelections

This transaction enables the policy selections for a registered provider to be set based on the policy definitions defined for the selections. The selection will be validated according to the policy definition.

Code Example 147: Set Policy Selections

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <SetPolicySelectionsRequest>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>option1</OptionName>
            <Value>value1</Value>
         </SimpleOptionSelection>
      </OptionSelection>
   </SetPolicySelectionsRequest>
</ProviderAccountService>
```

4.4.8 UpdateContact

To be supplied.

4.4.9 UpdateProviderContacts

This transaction enables the provider contacts to be updated. The updating information includes first name, last name, address information, phone information and email. Contact role can't be updated.

Code Example 148: Update provider contacts.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <UpdateContactRequest>
      <ProviderContact>
         <ContactRole>billing contact</ContactRole>
         <ContactFirstName>AJ</ContactFirstName>
         <ContactLastName>PennyPacker</ContactLastName>
      </ProviderContact>
   </UpdateContactRequest>
</ProviderAccountService>
```

4.4.10 UpdateProviderInformation

This transaction enables provider information, currently only the organization information, to be updated.

Code Example 149: Update provider information.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <UpdateProviderInformationRequest>
      <ProviderInformation>
         <OrganizationName>NASA DAAC</OrganizationName>
      </ProviderInformation>
   </UpdateProviderInformationRequest>
</ProviderAccountService>
```

4.4.11 GrantProviderAccess

This transaction grants the specified users access to a provider. The user who grants access must currently have access to the provider they are interested in. Also, that user must either have set their provider context to that provider or they must only have access to just that one provider.

Code Example 150: Grant provider access.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <GrantProviderAccessRequest>
      <UserName>User101</UserName>
   </GrantProviderAccessRequest>
</ProviderAccountService>
```

4.4.12 RevokeProviderAccess

This transaction revokes access to a provider from the specified users. The user who revokes access must currently have access to the provider they are interested in. Also, that user must either have set their provider context to that provider or they must only have access to just that one provider.

Code Example 151: Revoke provider access.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <RevokeProviderAccessRequest>
      <UserName>User101</UserName>
   </RevokeProviderAccessRequest>
</ProviderAccountService>
```

4.4.13 *Provider Inspect Functions*

Inspect functions allow the provider to view information about their holdings stored in ECHO. Providers can use this information to validate their holdings. Since these transactions are for providers, ECHO imposes no visibility restrictions on the datasets (in this context, "dataset" is synonymous with "collection") and granules.

Note: The ECHO system does not perform the validation. Providers validate their holdings based on the results returned from these transactions.

4.4.13.1 InspectHoldings

This transaction returns the following information:

Names of the provider's datasets,

Total number of granules in each dataset.

Code Example 152: Inspect dataset inline return.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
                              <InspectProviderHoldingsRequest/>
</ProviderAccountService>
```

4.4.13.2 InspectDataset

This transaction returns the following information:

Granule UR.

Provider's last update time. Time ranges may be specified based on the provider's last update time.

ECHO's last update time.

Providers must select the method by which they would like to receive the results from this transaction:

Inline: results are returned directly through the API.

E-mail.

FTP. ECHO returns an FTP URL for the provider to download.

Notes:

ECHO imposes a 1,000,000 granule size limit on inline and e-mail results.

The FTP URL expires two (2) days after ECHO receives the request.

Please allow ECHO time to assemble the FTP file for large datasets.

The code examples below demonstrate how to select the method by which they receive results.

Code Example 153: Inspect dataset inline return.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <InspectDatasetRequest>
      <DataSetID>ASTER Digital Elevation Model V003</DataSetID>
      <ProviderLastUpdateRange>
         <StartTime>2001-01-10</StartTime>
         <StopTime>2002-01-10</StopTime>
      </ProviderLastUpdateRange>
      <InspectionReturnType>
         <INLINE/>
      </InspectionReturnType>
   </InspectDatasetRequest>
</ProviderAccountService>
```

Code Example 154: Inspect dataset e-mail return.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <InspectDatasetRequest>
      <DataSetID>ASTER Digital Elevation Model V003</DataSetID>
      <ProviderLastUpdateRange>
         <StartTime>2001-01-10</StartTime>
         <StopTime>2003-01-10</StopTime>
      </ProviderLastUpdateRange>
      <InspectionReturnType>
         <EMAIL/>
      </InspectionReturnType>
      <EmailString>wu@gst.com</EmailString>
   </InspectDatasetRequest>
</ProviderAccountService>
```

Code Example 155: Inspect dataset FTP return.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderAccountService PUBLIC "-//ECHO ProviderAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderAccountService.dtd">
<ProviderAccountService>
   <InspectDatasetRequest>
      <DataSetID>ASTER Digital Elevation Model V003</DataSetID>
```

```
        <ProviderLastUpdateRange>
            <StartTime>2001-01-10</StartTime>
            <StopTime>2003-01-10</StopTime>
        </ProviderLastUpdateRange>
        <InspectionReturnType>
            <FTP/>
        </InspectionReturnType>
        <EmailString>wu@gst.com</EmailString>
    </InspectDatasetRequest>
</ProviderAccountService>
```

4.5 Addresses

Like regular user accounts, provider accounts contain multiple address entities. A provider account may have separate address entities associated with customer support locations, POP locations, etc. For more information about the address entity, see Section 3.1.2.

4.6 Emails

Like regular user accounts, email address is associated with a person, here the provider contact. Each provider contact owns one email address.

4.7 Provider Policy Update

As described in *ECHO Basics*, options provide a mechanism for customizing a provider's needs within the ECHO system. As with the user options (see *Client Interface*), the provider may set various option values to customize their sessions with ECHO. Providers will also be able define Options to describe required and optional properties that may affect the behavior of various services when their data is involved. There are currently three provider communication options defined for providers:

- submit
- quote
- cancel

There is another communication option that ECHO uses – validate. ECHO uses settings from the submit option to validate.

4.8 Metadata Ingest and Update

Metadata ingest is a backend process in the ECHO system. The main task for metadata ingest process is loading bulk metadata and updating the database. To improve the performance and efficiency of input data handling, the metadata ingest process uses FTP as the data transmission interface for the data provider metadata submission. The metadata input must be in XML format and conform to the ECHO DTDs or BMGT DTDs. The following sections discuss ECHO DTDs defined for collection metadata input and granule metadata input.

Each data provider is assigned a database account to host his or her metadata. Data providers are referred to with their own provider name - a unique identity in the ECHO system. The provider name is determined by agreement between the data provider and the ECHO Operations team. Collection, granule, and browse metadata are presented in XML files using either ECHO metadata DTDs or ECS BMGT metadata DTDs. The ECHO metadata DTD is published at the following URLs:

http://www.echo.eos.nasa.gov/dtd/v5.5/collectioningest.dtd

http://www.echo.eos.nasa.gov/dtd/v5.5/granuleingest.dtd.

If a data provider generates the XML file using their own DTD, then a provider's DTD must be converted to ECHO's DTD before sending into ECHO.

Since metadata update is part of ingest, the update applies to the same provider schema. The difference is that the provider has to follow ECHO's metadata update DTD, and some other rules for the metadata update XML files.

4.9 Metadata Update DTD

Currently, ECHO only handles granule level online access URL (delete, update, insert) and QA Flags (update) metadata update. Collection level metadata updates are ignored.

Code Example 156: Metadata update DTD.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v3.5 NT (http://www.xmlspy.com) by Keith Wichmann (Global Science and Technology, Inc.) -->
<!ELEMENT ProviderAccountService (UpdateMetadata)>
<!--UpdateMetadata can update a single collection, multiple collections, a single granule, or multiple granules in one
transaction.  Each update allows the addition of new metadata-->
<!ELEMENT UpdateMetadata (Collection*, Granule*)>
<!ELEMENT Collection (Target+, (Add | Update | Delete)+)>
<!ELEMENT Granule (Target+, (Add | Update | Delete)+)>
<!-- Target+ allows the same change to be made to several different granules or collections simultaneously.  This is
especially useful for bulk deletions of OnlineURLs. -->
<!ELEMENT Target (ID, ProviderLastUpdateDateTime, SaveDateTimeFlag?)>
<!— SaveDateTimeFlag is the flag that allows echo to update the last update date time for Target. The default is SAVE -->
<!ELEMENT Add (QualifiedTag, MetadataValue)>
<!ELEMENT Update (QualifiedTag, MetadataValue)>
<!ELEMENT Delete (QualifiedTag+)>
<!ELEMENT QualifiedTag (#PCDATA)>
<!ELEMENT MetadataValue (#PCDATA)>
<!ELEMENT ProviderLastUpdateDateTime (#PCDATA)>
<!ELEMENT SaveDateTimeFlag (SAVE | DONTSAVE)>
<!ELEMENT SAVE EMPTY>
<!ELEMENT ID (#PCDATA)>
<!ELEMENT DONTSAVE EMPTY>
```

The QualifiedTag is a string that follows the XPath standard. It is used to indicate which item in the DTD representing ECHO's metadata model is to be manipulated (e.g., updated, inserted, or deleted). In the case of deleting all online URLs for granules or collections, the XPath need only look like "OnlineAccessURLs". An optional flag, SaveDateTimeFlag, allows the date/time timestamp that identifies when metadata was last updated to be changed to reflect new update date/time information.

4.10 Values for the Qualified Tag for Updates, Deletes, and Inserts

The following DTD excerpts list the acceptable tags used to indicate which field in the ECHO data model is to be updated.

Note: Not all fields in the data model are available for update as of ECHO Release 5.5.

Code Example 157: DTD excerpt for collection metadata.

```
<!ELEMENT OnlineAccessURLs (OnlineAccessURL +)>
<!ELEMENT OnlineAccessURL (URL, URLDescription?, MimeType)>
<!ELEMENT CollectionOnlineResources (OnlineResource +)>
<!ELEMENT OnlineResource (OnlineResourceURL, OnlineResourceDescription?, OnlineResourceType, OnlineResourceMimeType)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT URLDescription (#PCDATA)>
<!ELEMENT MimeType (#PCDATA)>
<!ELEMENT OnlineResourceURL(#PCDATA)>
```

```
<!ELEMENT OnlineResourceDescription(#PCDATA)>
<!ELEMENT OnlineResourceType (#PCDATA)>
<!ELEMENT OnlineResourceMimeType (#PCDATA)>
```

Code Example 158: DTD excerpt for granule metadata.

```
<!ELEMENT OnlineAccessURLs (OnlineAccessURL +)>
<!ELEMENT OnlineAccessURL (URL, URLDescription?, MimeType)>
<!ELEMENT GranuleOnlineResources (OnlineResource +)>
<!ELEMENT OnlineResource (OnlineResourceURL, OnlineResourceDescription?, OnlineResourceType, OnlineResourceMimeType)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT URLDescription (#PCDATA)>
<!ELEMENT MimeType (#PCDATA)>
<!ELEMENT OnlineResourceURL(#PCDATA)>
<!ELEMENT OnlineResourceDescription(#PCDATA)>
<!ELEMENT OnlineResourceType (#PCDATA)>
<!ELEMENT OnlineResourceMimeType (#PCDATA)>
```

OnlineAccessURLs are to be used only for the actual data. URLs to all other kinds of web pages (including metadata) must be sent as OnlineResources.

Table 4-1 identifies valid XPath values for the qualified tags.

Table 4-1: Valid XPath values for qualified tags.

Tag	XPath	Value
Granule URL Delete One	OnlineAccessURLs/OnlineAccessURL[URL=”old_url”]/URL	
Granule URL Delete All	OnlineAccessURLs	
Granule URL Update	OnlineAccessURLs /OnlineAccessURL[URL=”old_url”]/URL	New url
Granule URL Insert	OnlineAccessURLs /OnlineAccessURL/URL	url
Granule Online resource Delete One	GranuleOnlineResources/OnlineResource[OnlineResourceURL=”old_url”]/ OnlineResourceURL	
Granule Online resource URL Update	GranuleOnlineResources/OnlineResource[OnlineResourceURL=”old_url”] / OnlineResourceURL	New url
Granule Online resource type Update	GranuleOnlineResources/OnlineResource[OnlineResourceURL =”old_url”]/OnlineResourceType	New type
Granule Online resource mime type Insert	GranuleOnlineResources/OnlineResource[OnlineResourceURL =”old_url”]/OnlineResourceMimeType	Mime type
Granule Automatic QA Flag Update	MeasuredParameter/MeasuredParameterContainer [ParameterName=”p1”]/QAFlags/AutomaticQualityFlag	New flag
Granule Automatic QA Flag explanation Update	MeasuredParameter/MeasuredParameterContainer [ParameterName=”p1”]/QAFlags/AutomaticQualityFlagExplanation	New explanation

Tag	XPath	Value
Granule Operational QA Flag Update	MeasuredParameter/MeasuredParameterContainer [ParameterName="p1"]/QAFlags/OperationalQualityFlag	New flag
Granule Operational QA Flag explanation Update	MeasuredParameter/MeasuredParameterContainer [ParameterName="p1"]/QAFlags/OperationalQualityFlagExplanation	New explanation
Granule Science QA Flag Update	MeasuredParameter/MeasuredParameterContainer [ParameterName="p1"]/QAFlags/ScienceQualityFlag	New flag
Granule Science QA Flag explanation Update	MeasuredParameter/MeasuredParameterContainer [ParameterName="p1"]/QAFlags/ScienceQualityFlagExplanation	New explanation

4.11 Packaging/Shipping Options

A new provider needs to specify what ordering options are made available to clients when orders are placed. Although the metadata ingest DTD has a place to specify these options, it is not currently implemented. Providers should contact the ECHO Operations team to have order options configured in the ECHO system. Refer to Order Entry Service in the *ECHO Client Interface Guide*, for a discussion of possible options.

4.12 Spatial Information

The ECHO system uses Oracle 9i to store the spatial information for collections and granules. It is important for data providers to prepare the spatial data appropriately to ensure that the spatial search will return correct results.

ECHO system accepts both Cartesian and Geodetic coordinate systems (WGS 84). A provider chooses a coordinate system based on the size and location of the spatial area covered.

With Oracle9i Cartesian coordinate system, the acceptable spatial data types include Point, Circle, Line, Bounding Box, and Polygon. With Oracle9i Geodetic coordinate system (WGS 84), the spatial data types supported include point, line, and polygon.

Table 4-2: Supported spatial data types.

Spatial data type	Cartesian	Geodetic	Notes
Point	✓	✓	
Circle	✓	×	Oracle spatial uses three points on the circumference of the circle. Any spatial data received as a circle expressed in center-latitude, center-longitude, and radians will not be stored as Oracle spatial data. Instead, it will be stored in a regular relational table.
Line	✓	✓	A line that has vertices across the International Date Line or across the poles will be considered invalid spatial data for flat Cartesian coordinate system.
Bounding box	✓	×	Since the Geodetic model does not support the bounding box type, ECHO system stores bounding box data as a four vertices polygon in flat Cartesian coordinate system for data process unification.
Polygon	✓	✓	Polygon vertices must be stored in order of vertex connection. The vertices should be sent in clockwise order. In addition to the order of the vertices, any consecutive vertices cannot have the same latitude and longitude, i.e., no repeating points. A line or a polygon that has vertices across the International Date Line or across

			the poles will be considered invalid spatial data for flat Cartesian coordinate system.

The Oracle spatial Geodetic model uses the shortest distance line to connect two vertices in order to construct the polygon area or the line. If there is not enough density for a set of vertices, then the line or the polygon area could be misinterpreted or the data could be considered invalid. Any polygon should not cover more then half of the earth.

To prepare the ECHO system to support polar search and Geodetic search, we store the spatial data in three possible types: point, line, and polygon. To avoid misinterpretation of a data provider's spatial data, the ECHO system will not manipulate any of the spatial input data. Data providers are responsible for the correctness and integrity of their spatial data. When spatial data is sent to ECHO, data providers must notify ECHO in advance as to what coordinate system is being used for the spatial data. Presumably one coordinate system will be used. Data providers should chose an appropriate data type for their spatial data. Data providers should also provide spatial data with appropriate density if using the Geodetic model. If any spatial data input is considered invalid by Oracle spatial, spatial search is precluded on the collection or granule.

By default, ECHO assumes the maximum spatial coverage area for any data provider is the whole Earth (-180 to 180 for longitude and –90 to 90 for latitude). The data provider should specify the maximum spatial coverage area otherwise.

The application must define the resolution for vertices' degree representation. If any two vertices' difference is less than the resolution, those two vertices will be considered identical, which might cause the spatial data become invalid for Oracle spatial. The data provider should provide the resolution for both latitude and longitude for their spatial data in advance.

Note: At this time, a spatial search is precluded on a collection or granule that uses a circle spatial coverage representation.

4.13 Temporal Information

The ECHO system does not require any specific date format. However, ECHO does require data providers to use only one date format for all data that reflects day/time information and that they use one of the conventional date formats. The data provider must notify the ECHO Operations team, in advance, which date format is to be used. The value of the temporal information such as collection's range – beginning date and range ending date etc. – must be GMT/UTC.

4.14 Product Specific Attributes

The ECHO system stores the product specific attributes including name and value exactly the way they are received. The ECHO system does not do a unification or mapping of the product specific attributes' name among the data providers. Searches against PSA name and value are case-insensitive.

The path/row information and cloud cover information can be singled out for fast searching. To facilitate a quick search on path/row for Landsat 7 data, providers can provide data in either way listed below:

Put the path/row data under the TAG <StartPath>, <EndingPath>, <StartRow>, and <EndingRow> for granules.

Notify the ECHO system of the PSA name for this set of PSA values in advance and send the information over as part of the PSAs. The ECHO system ingest procedure will single out the data and make the quick search upon path/row available.

To facilitate the quick search on the cloud cover information, the data provider should give the ECHO system the specific PSA name for this information in advance and send over the information as part of the PSAs. The ECHO system ingest procedure will single out this data and make the quick search upon the cloud cover available.

4.15 New Items vs. Update Items

In the ECHO DTDs for granule or collection metadata, there are TAGs of <Granules>, <NewGranules>, and <UpdateGranules> as well as TAGs of <Collections>, <NewCollections>, and <UpdateCollections>. Those tags indicate the action expected for the given set of granule or collection input data. However, to keep the data integrity for both data provider and the ECHO system, we expect a complete set of metadata for the granule or collection involved for insertion or updating to be sent over. Thus, the ECHO system can process the input data in a unified fashion. When processing a granule or collection, the ECHO system first checks to see if the item already exists. If the item already exists in the system, all the data associated with this item will be deleted before the new data is inserted into the ECHO database. The item identification is used to search for existing data. For a granule, the GranuleUR is assumed to be a unique identification for the data provider, while for collections the short name plus version number are assumed to be a unique identification for the data provider. For those who use BMGT DTD to generate the metadata, the tags of InsertTime and LastUpdate for the granule or collection are used to indicate the action of insertion or updating. The ECHO system applies the same principle to deal with the insertion or update of the granule or collection when processed against the XML metadata input. Furthermore, when updating a collection or a granule, only the one with most recent update time will be stored in the ECHO database. Any collection/granule received from the provider with a last update date earlier then the records in the ECHO database is ignored.

4.16 Delete Items

Data providers will instruct ECHO to delete collections or granules by placing the collections or granules under the tag of <DeleteCollections> or <DeleteGranules>. For those who are using the BMGT DTD to generate the XML metadata file, a non-empty DeleteTime data entry is the indication to delete this item. Only the item's identification is used to process the deletion of the item and all the data associated with this item. The deleted items' identification and deletion date will be kept in the ECHO database for data history auditing purpose. The only way to re-install the collections or granules in the ECHO system is to re-submit the metadata for those items to ECHO for insertion.

4.17 Miscellaneous

The DTD does not provide descriptions of certain data characteristics such as data type, length of field, etc. This additional information is available in the ECHO data dictionary located at http://www.echo.eos.nasa.gov/documents/ECHO_DTDTAG_DICTIONARY1.xls. If data received does not meet the specifications described in the data dictionary, those data records will be ignored.

For any repetitive item identification in the metadata input, only one item bearing the most recent update date with its complete information will be ingested into the database. The rest of the items and their associated information will be ignored even if they contain different data information.

The automated ingest process will generate an ingest summary report for each provider with the information of ingest start/stop time, data files received with the file size respectively, number of collection/granule processed, etc. The ingest report will be sent to designated provider personnel (specified as the ProviderContact when the provider registered with ECHO) via email in XML format.

4.18 Browse Image Files and Browse Metadata

When generating browse metadata xml files, browse image files and the corresponding browse metadata should be part of the output. ECHO will allocate the storage of the browse binary files, build the browse image URL, and update the database to associate the browse URL to its item record. The browse binary files indicated in the browse metadata XML file are expected to match the file size and name of the browse binary file. If the information provided in the browse metadata file is incorrect or incomplete, then the browse ingest process will not be activated.

NOTE: Browse ingest processing is currently implemented only with the BMGT Browse DTD. ECHO ingest does not currently support browse ingest via the ECHO Collection DTD and Granule DTD.

4.19 Notification of Metadata Transmission

No specific notification of metadata transmission is required. The auto ingest process monitors the file transmission and file transmission completion.

4.20 Online Data Access URL vs. Online Resources URL

There could be many online information available for a collection or granule such as Guides etc.. For certain granule or collection, the raw data are made available via online FTP site or online downloadable site. The URL information associated with a collection or granule could be sent to ECHO and made available to the user. ECHO chooses to store the URL information regarding directly accessible raw data separately with any other URL regarding other aspect of the collections or granules. ECHO require data provider send the URL using the TAG <OnlineAccessURLs> only for the URL that provides direct access of the raw data. Any other type of URLs should be sent using the TAG <CollectionOnlineResources> or <GranuleOnlineResources>.

4.21 Creating ECHO Compatible XML Files

ECS created the BMGT to generate XML metadata input files by extracting the data from the Sysbase-based database using the ECS schema. Many current ECHO data providers use BMGT. BMGT generates the XML input file based on four DTDs: collection metadata DTD, granule metadata DTD, browse metadata DTD, and valid metadata DTD. Currently, ECHO processes all the input data except valids. ECHO has its own DTDs defined for collection and granule metadata. ECHO ingest processes XML metadata files compliant to either the BMGT DTD or the ECHO DTDs. For collection XML input files generated by BMGT, the ECHO ingest process uses a proxy to convert the file into an ECHO collection DTD-conformant XML format before ingest. Data providers who do not use BMGT should generate the metadata XML input files using ECHO DTDs.

4.21.1 ECHO Collection DTD

For complete ECHO Collection DTD, please see http://www.echo.eos.nasa.gov/dtd/v5.5/collectioningest.dtd.

4.21.1.1 Basic Information of Collection

The basic elements for collection are: Short Name; Version Number; Long Name; Collection Description; Version Description; Revision Date; Suggested Usage; Processing Center; Archive Center; Citation For External Publication; Collection State; Maintenance Update; Processing Level; Access Constraints; Insert Day Time; and Last Update Day Time.

The short name and version number combination uniquely identifies a collection. Besides the short name and version number, ECHO also forms a dataset ID to uniquely identify a collection for the provider according to the rule provided by the data provider.

The Insert Day Time is the day and time the collection metadata was inserted in the provider's database. The Last Update Day Time is the day and time the collection metadata gotten update in the provider's database.

4.21.1.2 Temporal

Temporal information represents the time that the collection's data were collected. Three types of temporal information expression are used to present the temporal information associated with the collection. The types are: Single day time, Range day time, and Periodic day time. A collection could have more then one type of temporal associated. This information is essential search criteria for collection.

Code Example 159: Expression of temporal information (range day time).

```
<Temporal>
   <TimeType>UTC </TimeType>
   <DateType>Gregorian </DateType>
   <TemporalRangeType>Continuous Range</TemporalRangeType>
   <PrecisionofSeconds>1</PrecisionofSeconds>
   <EndsatPresentFlag>Y</EndsatPresentFlag>
   <RangeDateTime>
      <RangeBeginningDate>1998-01-01 00:00:00.0</RangeBeginningDate>
      <RangeBeginningTime>00:00:00.000000</RangeBeginningTime>
      <RangeEndingDate>1998-01-01 00:00:00.0</RangeEndingDate>
      <RangeEndingTime>00:00:00.000000</RangeEndingTime>
   </RangeDateTime>
</Temporal>
```

Temporal information must be submitted using UTC time and Gregorian day type.

The format of the date expression in this example is: yyyy-mm-dd hh24:mi:ss.ms(1) – four digits for year, two digits for month, two digits for day, two digits for hour in 24 hour system, two digits for minute, and two digits for second with one digit for precision of the second. Providers might express their date information in a different format and ECHO will process the information accordingly. However, once the format is defined and agreed upon between ECHO and the data provider, then all the date information coming from the data provider should follow the format. Otherwise, ECHO will not be able to process the information that will result an NULL entry for the information.

4.21.1.3 Spatial

Spatial data is the spatial area that the collection's data covers. This is an essential piece of information for collection search criteria. For detailed information on handling and expression of spatial information, please refer to section 4.12.

4.21.1.4 Sources and Sensors

ECHO adopts a layered representation of sources and sensors information for the collection. A source and sensor layer is defined as:

Platform->* Instrument->* Sensor

A collection could be associated with 0 or more platforms; each platform could contain 0 or more instruments, and each instrument could contain 0 or more sensors. There might be characteristic parameters associated with the platforms, instruments, and or sensors for the collection. In addition to the characteristic parameters, there might be operational mode associate to instruments. When the data provider does not have the platform concept with their collection, the input for platform short name should be "No Platform Associated". When the data provider does not have the instrument concept for their collection, the input for instrument short name should be "No Instrument Associated".

Code Example 160: Full platform/instrument/sensor description.

```
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <Instrument>
      <InstrumentShortName>MODIS</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
      <OperationMode>Operation Mode</OperationMode>
   </Instrument>
</Platform>
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <PlatformLongName>First EOS Polar Orbiting Satellite, 10:30 AM Descending Equator Crossing</PlatformLongName>
   <PlatformType>Spacecraft</PlatformType>
   <PlatformCharacteristic>
      <PlatformCharacteristicName>EquatorCrossingTime</PlatformCharacteristicName>
      <PlatformCharacteristicDescription>Local time of the equator crossing and direction (ascending or 
descending)</PlatformCharacteristicDescription>
      <PlatformCharacteristicDataType>varchar </PlatformCharacteristicDataType>
      <PlatformCharacteristicUnit>Local Mean Time</PlatformCharacteristicUnit>
      <PlatformCharacteristicValue>10:30, descending</PlatformCharacteristicValue>
   </PlatformCharacteristic>
   <Instrument>
```

```
      <InstrumentShortName>MODIS</InstrumentShortName>
      <InstrumentLongName>Moderate-Resolution Imaging Spectroradiometer</InstrumentLongName>
      <InstrumentTechnique>Imaging Spectroradiometry</InstrumentTechnique>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 161: No platform association.

```
<Platform>
   <PlatformShortName>No Platform Associated</PlatformShortName>
   <Instrument>
      <InstrumentShortName>MODIS</InstrumentShortName>
      <InstrumentLongName>Moderate-Resolution Imaging Spectroradiometer</InstrumentLongName>
      <InstrumentTechnique>Imaging Spectroradiometry</InstrumentTechnique>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 162: No instrument association.

```
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <PlatformLongName>First EOS Polar Orbiting Satellite, 10:30 AM Descending Equator Crossing</PlatformLongName>
   <PlatformType>Spacecraft</PlatformType>
   <PlatformCharacteristic>
      <PlatformCharacteristicName>EquatorCrossingTime</PlatformCharacteristicName>
      <PlatformCharacteristicDescription>Local time of the equator crossing and direction (ascending or
descending)</PlatformCharacteristicDescription>
      <PlatformCharacteristicDataType>varchar </PlatformCharacteristicDataType>
      <PlatformCharacteristicUnit>Local Mean Time</PlatformCharacteristicUnit>
      <PlatformCharacteristicValue>10:30, descending</PlatformCharacteristicValue>
   </PlatformCharacteristic>
   <Instrument>
      <InstrumentShortName>No Instrument Associated</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 163: Sensor only.

```
<Platform>
   <PlatformShortName>No Platform Associated</PlatformShortName>
   <Instrument>
      <InstrumentShortName>No Instrument Associated</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

The requirement for platform/instrument/sensor entry is compliant with GCMD standard valid. For GCMD standard sources/sensor valid, please view http://gcmd.gsfc.nasa.gov/Resources/valids/index.html

4.21.1.5 Keywords

ECHO supports three kinds of keyword associations for collections. Those are: discipline keywords, spatial keywords, and temporal keywords. For discipline keywords and spatial keywords, the requirement for keywords entry is compliant with GCMD keywords standard. For GCMD keywords standard, please see http://gcmd.gsfc.nasa.gov/Resources/valids/index.html

4.21.1.6 Product Specific Attributes

Product Specific Attributes (PSAs) (or referred in the DTD as Additional Attributes) are parameters that further describe the collection. These are important search criteria for the collection. ECHO supports collection-level PSA definition. No PSA values are associated with a collection at this time.

Code Example 164: Using product specific attributes.

```
<AdditionalAttributes>
   <AdditionalAttributeDataType>varchar </AdditionalAttributeDataType>
   <AdditionalAttributeDescription>Version of the process software that generated the
product.</AdditionalAttributeDescription>
   <AdditionalAttributeName>PROCESSVERSION</AdditionalAttributeName>
</AdditionalAttributes>

<AdditionalAttributes>
   <AdditionalAttributeDataType>String</AdditionalAttributeDataType>
   <AdditionalAttributeDescription>Estimated RMS error in geolocation</AdditionalAttributeDescription>
   <AdditionalAttributeName>GEO_EST_RMS_ERROR</AdditionalAttributeName>
</AdditionalAttributes>

<AdditionalAttributes>
   <AdditionalAttributeDataType>String</AdditionalAttributeDataType>
   <AdditionalAttributeDescription>Flag set (to 0) if science_state, the L1A engineering data flag that indicates the
Normal/Test configuration of the MODIS instrument, was set for at least one scan in the
granule.</AdditionalAttributeDescription>
   <AdditionalAttributeName>SCI_STATE</AdditionalAttributeName>
</AdditionalAttributes>

<AdditionalAttributes>
   <AdditionalAttributeDataType>String</AdditionalAttributeDataType>
   <AdditionalAttributeDescription>Flag set (to 0) if science_abnormal, the L1A engineering data ground-set flag that
indicates potentially abnormal science data due to things other than MODIS (such as maneuvers, data link, etc.), was set
for at least one scan in the granule.</AdditionalAttributeDescription>
   <AdditionalAttributeName>SCI_ABNORM</AdditionalAttributeName>
</AdditionalAttributes>

<AdditionalAttributes>
   <AdditionalAttributeDataType>int </AdditionalAttributeDataType>
   <AdditionalAttributeDescription>The number of the granule for the day starting at midnight. Example: The granule from
00:00:00 to 00:00:05 will be granule number 1. The granule from 00:01:00 to 00:01:05 will be granule number
13</AdditionalAttributeDescription>
   <AdditionalAttributeName>GRANULENUMBER</AdditionalAttributeName>
</AdditionalAttributes>
```

4.21.1.7 Other Descriptive Information

Other information associated with a collection includes:

- Computer Science Data Type (CSDT);
- Contact information;
- Campaign information;
- Association;
- Browse;
- Online URL;
- Coordinate system and planar information for collection's raw data.

4.21.2 ECHO Granule DTD

For complete ECHO granule DTD, please view http://www.echo.eos.nasa.gov/dtd/v5.5/granuleingest.dtd.

4.21.2.1 Granule/Collection Association

Earth Science metadata represented by ECHO are based on granule. Every granule has to belong to a primary collection. In addition to its primary collection, a granule could be included in the other collections. A collection could contain 0 to more then one granules. The primary collection/granule association information is provided via granule metadata input.

Code Example 165: Associating granules with collections.

```
<GranuleURMetaData>
   <GranuleUR>SC:MODMGGAD.001:81677</GranuleUR>
   <DbID>81677</DbID>
   <InsertTime>2001-09-20 11:43:31.996</InsertTime>
   <LastUpdate>2001-09-20 11:43:31.996</LastUpdate>
   <CollectionMetaData>
      <ShortName>MODMGGAD</ShortName>
      <VersionID>1</VersionID>
   </CollectionMetaData>
……
</GranuleURMetaData>
```

Where collection's short name and version ID combined uniquely identifies the primary collection that the granule belongs to. The granule will be deleted when its primary collection is deleted. Many information associated with the granule are defined or restricted by its primary collection.

4.21.2.2 Basic Information of Granule Including Temporal

The basic information of the granule includes: Granule UR (granule universal reference ID); Primary collection; Data file size; Processing information; Local granule reference; Day night Flag; PGE version; Temporal information; Access Constraint; and Dates associated with production and metadata update upon provider's database.

The access constraint indicates whether or not there is any access constraint applying when the granule data goes to public. If this constraint is not explicitly indicated for the granule, granule inherits its primary collection's access constraint.

Two types of temporal information represent the time when granule data were collected. One type is single day/time and the other type is range day/time. A granule's temporal should be in the range of it primary collection's temporal and should refer to the same temporal system (UTC/Gregorian for specific). Since the temporal system definition is provided by its primary collection, in granule's metadata input, the temporal system definition is not present.

Code Example 166: Temporal information is an essential piece of metadata for a granule search.

```
<GranuleURMetaData>
   <GranuleUR>SC:MODMGGAD.001:81677</GranuleUR>
   <DbID>81677</DbID>
   <InsertTime>2001-09-20 11:43:31.996</InsertTime>
   <LastUpdate>2001-09-20 11:43:31.996</LastUpdate>
   <CollectionMetaData>
      <ShortName>MODMGGAD</ShortName>
      <VersionID>1</VersionID>
   </CollectionMetaData>
   <ECSDataGranule>
      <SizeMBECSDataGranule>36.332</SizeMBECSDataGranule>
      <ReprocessingPlanned>further update is anticipated</ReprocessingPlanned>
      <ReprocessingActual>reprocessed</ReprocessingActual>
      <LocalGranuleID>MODMGGAD.A2001149.h08v07.003.hdf</LocalGranuleID>
      <DayNightFlag>Day </DayNightFlag>
      <ProductionDateTime>2001-07-24 12:30:03.0</ProductionDateTime>
      <LocalVersionID>3.0.2</LocalVersionID>
   </ECSDataGranule>
   <PGEVersionClass>
      <PGEVersion>3.0.2 </PGEVersion>
   </PGEVersionClass>
   <RangeDateTime>
      <RangeEndingTime>17:20:00.000000</RangeEndingTime>
      <RangeEndingDate>2001-05-29 00:00:00.0</RangeEndingDate>
      <RangeBeginningTime>17:10:00.000000</RangeBeginningTime>
      <RangeBeginningDate>2001-05-29 00:00:00.0</RangeBeginningDate>
   </RangeDateTime>
</GranuleURMetaData>
```

4.21.2.3 Spatial

The spatial data is the spatial area that granule's data covers. This is an essential piece of information for granule search criteria. The spatial coverage area for a granule should be within the spatial coverage area of its primary collection. Detailed handling and expression of spatial information please refer section 4.4.14.

4.21.2.4 Sources and Sensors

ECHO adopts a layered representation of sources and sensors information for the granule. A source and sensor layer is defined as:

Platform->* Instrument->* Sensor

A granule could be associated with 0 to more then one platforms, each platform could contain 0 to more then one instruments, and each instrument could contain 0 to more then one sensors. There won't be characteristic parameters associated with the platforms and instruments. Granule's platforms' and instruments' characteristic were defined by its primary collection. Granule could define sensor level characteristic value for the characteristic parameters defined by its primary collection. In addition to the characteristic parameters, there might be operational mode associate to instruments independent of its primary collection. When the data provider does not have the platform concept with their granule, the input for platform short name should be "No Platform Associated". When the data provider does not have the instrument concept for their granule, the input for instrument short name should be "No Instrument Associated". Although in the current version of ECHO granule DTD, there is the entry for platform/instrument characteristics and sensor level characteristics definition, following the rules stated above for granule sources/sensor information integrity is strongly recommended.

Code Example 167: Full platform/instrument/sensor description.

```
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <Instrument>
      <InstrumentShortName>MODIS</InstrumentShortName>
      <OperationMode>Operation Mode</OperationMode>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 168: No platform association.

```
<Platform>
   <PlatformShortName>No Platform Associated</PlatformShortName>
   <Instrument>
      <InstrumentShortName>MODIS</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 169: No instrument association.

```
<Platform>
   <PlatformShortName>Terra</PlatformShortName>
   <Instrument>
      <InstrumentShortName>No Instrument Associated</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

Code Example 170: Sensor only.

```
<Platform>
   <PlatformShortName>No Platform Associated</PlatformShortName>
   <Instrument>
      <InstrumentShortName>No Instrument Associated</InstrumentShortName>
      <Sensor>
         <SensorShortName>MODIS</SensorShortName>
         <SensorLongName>Cross-track Scanning Radiometer</SensorLongName>
         <SensorTechnique>Radiometry</SensorTechnique>
      </Sensor>
   </Instrument>
</Platform>
```

The platform, instrument, and sensor referred to by the granule is restricted by its primary collection source/sensor reference. The requirement for platform/instrument/sensor entry is compliant with GCMD standard valid. For GCMD standard sources/sensor valid, please view http://gcmd.gsfc.nasa.gov/Resources/valids/index.html

4.21.2.5 Product Specific Attributes

The Product Specific Attributes (PSA) is a group of parameters that further describes the granule with specific value. These are important search criteria for the granule. The granule's PSAs should be bound with the PSAs defined for its primary collection. A granule could either reference a PSA or reference a PSA with a value.

Code Example 171: Granules and PSAs.

```
<GranuleURMetaData>
…..
<PSAs>
   <PSA>
      <PSAName>QAPERCENTNOTPRODUCEDOTHER</PSAName>
   </PSA>

   <PSA>
      <PSAName>QAPERCENTNOTPRODUCEDCLOUD</PSAName>
   </PSA>

   <PSA>
```

```
      <PSAName>QAPERCENTGOODQUALITY</PSAName>
      <PSAValue>100</PSAValue>
   </PSA>

   <PSA>
      <PSAName>QAPERCENTOTHERQUALITY</PSAName>
      <PSAValue>0</PSAValue>
   </PSA>
</PSAs>
……
</GranuleURMetaData>
```

Although there is an entry in the ECHO granule DTD to allow PSA definition at the granule level, it is not recommended to do so.

4.21.2.6 Measured Parameters

Measured parameters are associated with a granule only and are important information for the granule. For some providers, the value of certain measured parameters decides the visibility of the granule.

Code Example 172: Measured parameters.

```
<GranuleURMetaData>
……
<MeasuredParameter>
   <MeasuredParameterContainer>
      <ParameterName>MODMGGAD</ParameterName>
      <QAStats>
         <QAPercentMissingData>0</QAPercentMissingData>
      </QAStats>
      <QAFlags>
         <AutomaticQualityFlag>Passed</AutomaticQualityFlag>
         <AutomaticQualityFlagExplanation>output file is created and good</AutomaticQualityFlagExplanation>
         <ScienceQualityFlag>Not Investigated</ScienceQualityFlag>
         <ScienceQualityFlagExplanation>See http://modland.nascom.nasa.gov/QA_WWW/release.html for the Science QA status of
this product.</ScienceQualityFlagExplanation>
      </QAFlags>
   </MeasuredParameterContainer>
</MeasuredParameter>
……
</GranuleURMetaData>
```

4.21.2.7 Orbital Information

Another piece of interesting information for a granule is the information associated with the orbital.

Code Example 173: Orbital information.

```
<GranuleURMetaData>
……
<OrbitCalculatedSpatialDomain>
   <OrbitCalculatedSpatialDomainContainer>
      <OrbitNumber>7694</OrbitNumber>
      <EquatorCrossingLongitude>-100.187</EquatorCrossingLongitude>
```

```
        <EquatorCrossingDate>2001-05-29 00:00:00.0</EquatorCrossingDate>
        <EquatorCrossingTime>17:19:38.910554</EquatorCrossingTime>
    </OrbitCalculatedSpatialDomainContainer>
</OrbitCalculatedSpatialDomain>
……
</GranuleURMetaData>
```

4.21.2.8 Other Descriptive Information

The other granule metadata includes: Campaign information, Browse, Online URL, and Processing/QA/Input historical information. A granule could associate with 0 to many browse files and a browse file could be referenced by more than one granule.

4.21.3 Translation Process

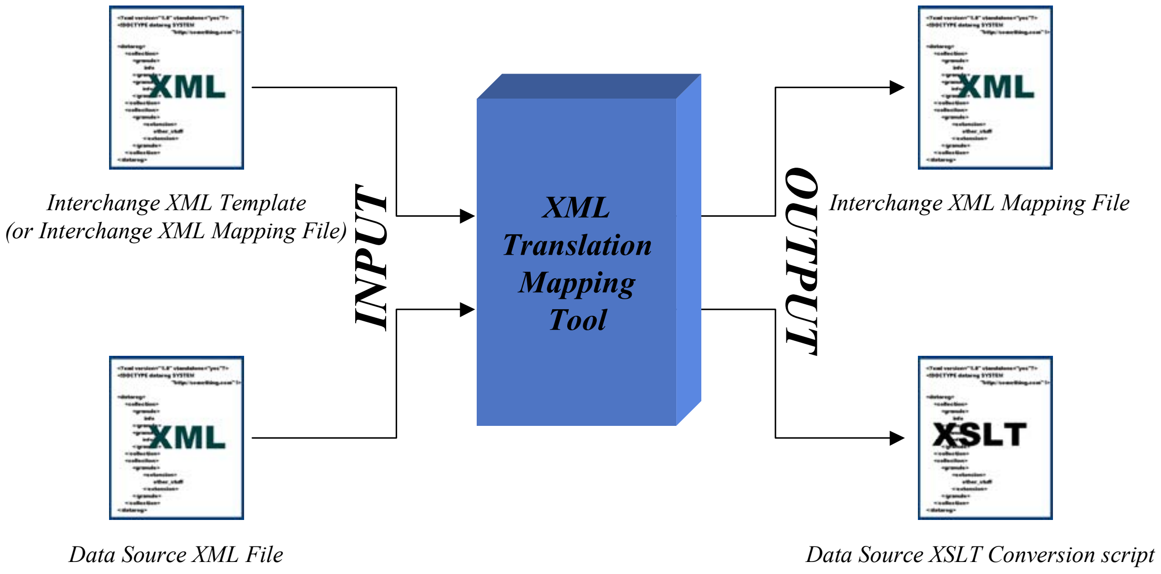

XML
Interchange XML Template
(or Interchange XML Mapping File)
XML
Data Source XML File
INPUT
XML Translation Mapping Tool
OUTPUT
XML
Interchange XML Mapping File
XSLT
Data Source XSLT Conversion script

Figure 4-1: XML translation process.

Using a wizard-like interface, the XML Translation Mapping Tool helps you create complex mappings between the elements in your Data Source XML file and the relevant elements in the Interchange XML Template file.

4.21.4 Metadata Mapping/Ingest Process

Metadata input files generated using BMGT or the ECHO DTD are accepted and processed directly by the ingest process. Metadata input files conforming to other DTDs require an XML-to-XML conversion before being submitted to the ECHO ingest process. The provider is strongly recommended to perform the conversion. Contact the ECHO Operations team for more information.

When the ECHO ingest process detects potential input files in a provider's FTP input directory, the ingest process will then examine and validate these files. Each file will be checked to see if it is a properly formed XML file by examining the first line of text, which must contain the <!xml....> declaration. If this line is not present, the input file will be rejected. Next, the ingest process will verify that the file is located in the appropriate directory. For example, if a granule XML file is placed in the collection FTP input directory the input file will be rejected. Finally, the ingest process will validate the input file with its corresponding DTD. If this validation fails, the input file will be rejected. Whenever an input file is rejected, an error will be recorded and sent to the provider via the pre-configured provider contact email address. The same information will be emailed to the ECHO OPS staff as well.

Once an input file has been successfully validated, the ECHO ingest process will load the data into the ECHO database and update the operational database tables. The data will also then be made available for public search and an ingest summary email will be sent to the provider via the pre-configured provider contact email address. The ingest summary information will be emailed to the ECHO OPS staff as well.

Additional constraints are applied at ingest to maintain granule data integrity. These constraints are:

- Collection metadata must be in the ECHO database before any of that collection’s metadata is accepted in ECHO.
- Granules' PSA attributes have must be a sub-set of its associated collection's PSA attributes.
- Granules' platform/instrument/sensor configurations must be a sub-set of its associated collection's platform/instrument/sensor configurations.
- Granules' instrument operation modes must be a sub-set of its associated collection's instrument operation modes.
- Granules' analysis sources must be a sub-set of its associated collection's analysis sources.
- Granules' campaigns must be a sub-set of its associated collection's campaigns.
- A granule will be rejected if any of the above constraints are violated.

4.22 Spatial Representations, Coordinates & Projections

For a collection or granule, the spatial area coverage is a basic – and one of the most important – search criteria for earth science data, although this is not required data. There are some specific issues that need to be discussed to enable the user to submit the metadata correctly.

ECHO, currently, supports multiple spatial representations. Spatial data of different spatial representation are described differently, hence are stored and searched in different way. Each provider is allowed to have different spatial representation, however, each collection must define a single spatial representation for its granules. The following sections cover details on the spatial representations.

ECHO currently supports two coordinate systems for spatial data. Each data provider should use only one coordinate system when constructing the spatial area coverage for a collection or granule. There are some differences in terms of spatial data representation for those two coordinate systems.

4.22.1 Geometry Representations

Spatial data most commonly seen are described as certain geometry such as a polygon, a multi-polygon, or a line. They are stored as ORACLE spatial objects in the database to record its shape, spatial locations of corner points and spatial coordinate system used, Cartesian or Geodetic. Cartesian and Geodetic are considered as different spatial representations due to the difference in the manner in which the two components are connected. Straight lines connect corner points in the Cartesian system, while in the Geodetic system they are connected by great circle arcs. This makes their spatial coverages slightly different. Oracle requires specifying the coordinate system during data storage time. Therefore, data under different coordinate system are also searched differently.

4.22.2 Coordinate System

4.22.2.1 Cartesian Coordinate System

The Cartesian Coordinate System is a flattened coordinate system with longitude ranged from –180 to 180 degrees and latitude ranged from –90 to 90 degrees. The projected map is flattened and open along the International Date Line with North Pole and South Pole as top and bottom line respectively.

4.22.2.2 Geodetic Coordinate System

The Geodetic coordinate system is defined in angular (latitude and longitude) and is defined relative to spherical polar coordinate and Earth Geodetic datum. Oracle defines coordinate system following OGC standards. The Geodetic coordinate ECHO chose to support is WGS 84. It is defined as:

GEOGCS ["Longitude / Latitude (WGS 84)", DATUM ["WGS 84", SPHEROID ["WGS 84", 6378137.000000, 298.257224]], PRIMEM ["Greenwich", 0.000000], UNIT ["Decimal Degree", 0.01745329251994330]]

4.22.3 Data Types and Representation

4.22.3.1 Point

ECHO can receive, store, and support the search on spatial data representing one or more points. To send ECHO spatial point data in the metadata XML file, the information is expressed as follows.

Code Example 174: Single point.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Point>
         <PointLongitude>-123.948</PointLongitude>
         <PointLatitude>45.0664</PointLatitude>
      </Point>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

Code Example 175: Multiple points.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Point>
         <PointLongitude>-123.948</PointLongitude>
          <PointLatitude>45.0664</PointLatitude>
      </Point>
      <Point>
         <PointLongitude>-133.546</PointLongitude>
         <PointLatitude>45.0664</PointLatitude>
      </Point>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

4.22.3.2 Line

ECHO can receive, store, and search spatial data representing one or more lines. To send ECHO spatial line data in the metadata XML file, the information is expressed as:

Code Example 176: Single line.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Line>
         <Point>
            <PointLongitude>-123.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>-133.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
      </Line>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer >
```

Code Example 177: Multiple lines.

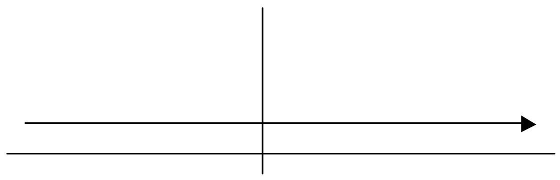
```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Line>
         <Point>
            <PointLongitude>-123.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>-133.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
      </Line>
      <Line>
         <Point>
            <PointLongitude>-123.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>-133.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>-143.546</PointLongitude>
            <PointLatitude>40.0664</PointLatitude>
         </Point>
      </Line>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer >
```

In the Cartesian coordinate system, the points are connected with a straight line on the plane in the order as listed. The line connects two points will never cross the International Date Line or Poles.

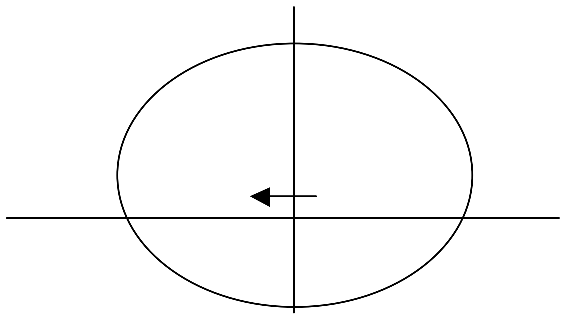
Code Example 178: Code to be interpreted in Cartesian and Geodetic systems.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Line>
         <Point>
            <PointLongitude>-173.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>173.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
      </Line>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

The figure below shows a line in the Cartesian coordinate system:



This same expression in the Geodetic coordinate system represents the line as:



In the Geodetic coordinate system, the line connects two points using great circle arc with shortest distance between the two points. The line could cross the International Date Line and Poles. If the same line indicated in the Cartesian coordinate system should be represented in the Geodetic coordinate system, the expression should be

Code Example 179: Adding density.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Line>
         <Point>
            <PointLongitude>-173.948</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>0.0</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
         <Point>
            <PointLongitude>173.546</PointLongitude>
            <PointLatitude>45.0664</PointLatitude>
         </Point>
      </Line>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer >
```

The additional point gives the line more density so that Oracle interoperates the data correctly. The appropriate density should be applied to other spatial type for Geodetic coordinate system as well.

4.22.3.3 Polygon

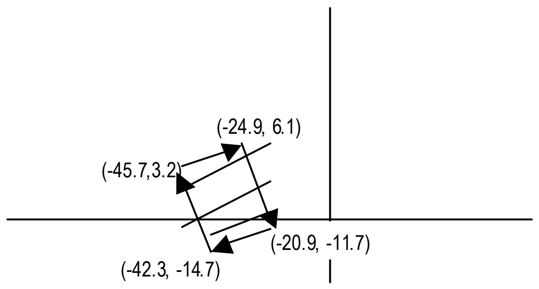
ECHO is capable of receiving, storing, and searching spatial data representing a polygon, a polygon with hole, and multiple polygons (any type – with hole or without hole). To send ECHO spatial polygon data in the metadata XML file, the information is expressed as:

Code Example 180: Single polygon.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-20.9342</PointLongitude>
                     <PointLatitude>-11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-42.3067</PointLongitude>
                     <PointLatitude>-14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-45.7985</PointLongitude>
                     <PointLatitude>3.198</PointLatitude>
                  </Point>
                  <Point>
                    <PointLongitude>-24.8982</PointLongitude>
                    <PointLatitude>6.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

A single polygon can only have one outer ring that represents the area surrounded within. The points are connected using a straight line on Cartesian coordinate system in the order listed. In Cartesian coordinate system, the points should be listed in the order of clockwise connection. The area should not cross the International Date Line or Poles. In the Geodetic coordinate system, the points are connected using great circle arc according to the shortest distance between the two points. The density concept is especially important for polygon representation in the Geodetic coordinate system. The polygon coverage could cross the International Date Line and/or poles in Geodetic coordinate system.

The spatial area covered by the above representation is:

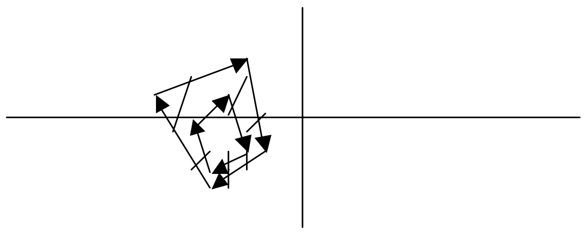


Code Example 181: Single polygon with hole.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-20.9342</PointLongitude>
                     <PointLatitude>-11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-42.3067</PointLongitude>
                     <PointLatitude>-14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-45.7985</PointLongitude>
                     <PointLatitude>3.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-24.8982</PointLongitude>
                     <PointLatitude>6.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
            <InnerRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-22.9342</PointLongitude>
                     <PointLatitude>-5.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-30.3067</PointLongitude>
                     <PointLatitude>-10.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-35.7985</PointLongitude>
                     <PointLatitude>1.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-30.8982</PointLongitude>
                     <PointLatitude>3.1665</PointLatitude>
                  </Point>
               </Boundary>
            </InnerRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

A single polygon with hole can only have one outer ring that represents the area surrounded within. A single polygon with hole can have multiple inner rings that represent holes. All the rules, restrictions, and discussions for the outer ring in both coordinate systems apply to inner ring. An inner ring should be completely contained by an outer ring.

The spatial area covered by above representation is:



Multiple Polygons:

A multiple polygon is the combination of polygons. Those single polygons could be with or without holes. Each single polygon expression should follow the same rules listed above. No two polygons may overlap each other.

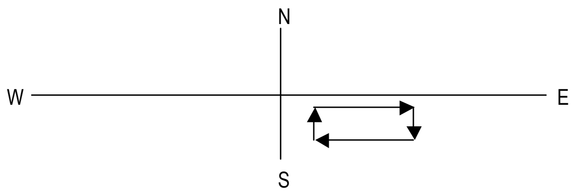
4.22.3.4 Bounding Box

ECHO is capable of receiving, storing, and supporting the search on spatial data representing a bounding box or multiple bounding boxes. To send ECHO spatial point data in the metadata XML file, the information is expressed as:

Code Example 182: Using a bounding box.

```
<HorizontalSpatialDomainContainer>
   <BoundingRectangle>
      <WestBoundingCoordinate>8.733</WestBoundingCoordinate>
      <NorthBoundingCoordinate>-7.4861</NorthBoundingCoordinate>
      <EastBoundingCoordinate>43.199501</EastBoundingCoordinate>
      <SouthBoundingCoordinate>-35.2617</SouthBoundingCoordinate>
   </BoundingRectangle>
</HorizontalSpatialDomainContainer>
<HorizontalSpatialDomain>
   <ZoneIdentifier/>
     <BoundingRectangle>
        <WestBoundingCoordinate><8.733></WestBoundingCoordinate>
        <NorthBoundingCoordinate><-7.4861></NorthBoundingCoordinate>
        <EastBoundingCoordinate><43.199501></EastBoundingCoordinate>
       <SouthBoundingCoordinate><-35.2617></SouthBoundingCoordinate>
     </BoundingRectangle>
</HorizontalSpatialDomain>
```

The spatial area coverage is:



ECHO stores a bounding box as a four pointed polygon, subject to the specifications and constraints described for polygons in section 4.22. In the Cartesian system, bounding boxes cannot cross the International Date Line or poles.

In the Geodetic coordinate system, bounding box coverage is not allowed to have an area greater than or equal to half the area of the Earth. If there is more than one bounding box listed, then ECHO stores them as multiple polygons which cannot overlap.

Note: Data providers that use the Geodetic coordinate system for their spatial data should be particularly cognizant of the Geodetic "half Earth" restriction described above. These data providers should use polygons rather than bounding boxes when describing their data.

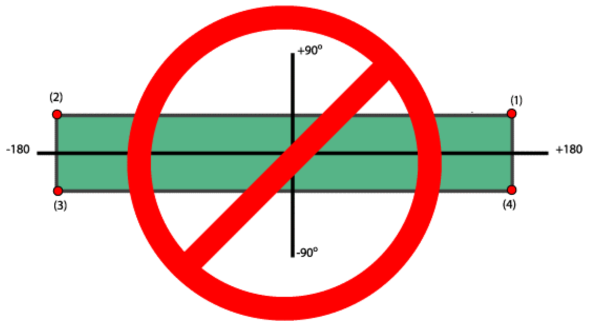
4.22.4 Invalid Spatial Representation

4.22.4.1 Polygon with points in counter-clockwise order

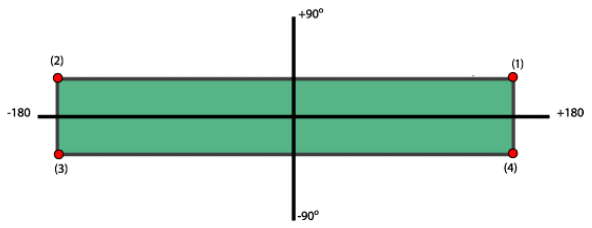
Code Example 183: Polygon with points in counter-clockwise order.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>170</PointLongitude>
                     <PointLatitude>30</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-170</PointLongitude>
                     <PointLatitude>30</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-170</PointLongitude>
                     <PointLatitude>-30</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>170</PointLongitude>
                     <PointLatitude>-30</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

This spatial area expression is considered invalid in the Cartesian coordinate system because of the order of the points shown on the plane. However, this same expression is considered valid in the Geodetic coordinate system, although the coverage will be interpreted differently.



Cartesian



Geodetic

Figure 4-2: This expression is invalid in the Cartesian coordinate system, but valid in the Geodetic coordinate system.

4.22.4.2 Twisted Polygon

Code Example 184: Twisted polygon.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-20.9342</PointLongitude>
                     <PointLatitude>-11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-42.3067</PointLongitude>
                     <PointLatitude>-14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-24.8982</PointLongitude>
                     <PointLatitude>6.1665</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-45.7985</PointLongitude>
                     <PointLatitude>3.198</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

This presentation shows a polygon as:

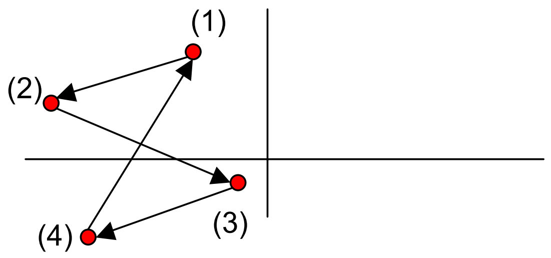


Figure 4-3: Twisted polygon.

In either coordinate system, this expression represents an invalid polygon.

4.22.4.3 Hole crosses over outer ring

Code Example 185: Hole crosses the outer ring.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-20.9342</PointLongitude>
                     <PointLatitude>-11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-42.3067</PointLongitude>
                     <PointLatitude>-14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-45.7985</PointLongitude>
                     <PointLatitude>3.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-24.8982</PointLongitude>
                     <PointLatitude>6.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
            <InnerRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-17.9342</PointLongitude>
                     <PointLatitude>-5.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-30.3067</PointLongitude>
                     <PointLatitude>-10.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-35.7985</PointLongitude>
                     <PointLatitude>1.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-10.8982</PointLongitude>
                     <PointLatitude>3.1665</PointLatitude>
                  </Point>
               </Boundary>
            </InnerRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

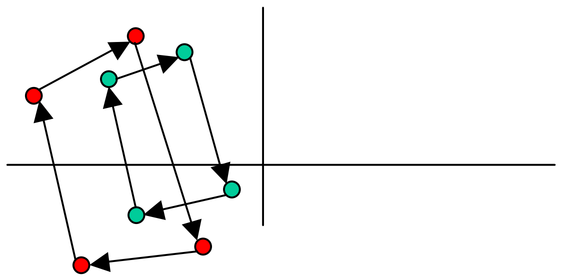


Figure 4-4: Hole crosses outer ring.

In either coordinate system, this represents an invalid spatial area.

4.22.4.4 Polygon Crosses International Date Line

Code Example 186: Polygon crosses the International Date Line.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>170.9342</PointLongitude>
                     <PointLatitude>11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-175.3067</PointLongitude>
                     <PointLatitude>14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-176.7985</PointLongitude>
                     <PointLatitude>-13.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>172.8982</PointLongitude>
                     <PointLatitude>-7.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

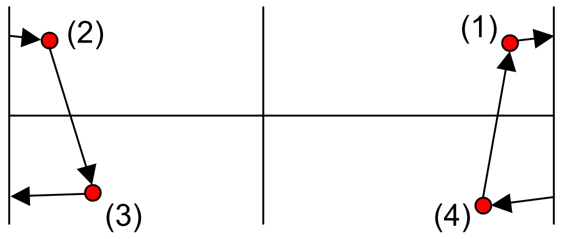
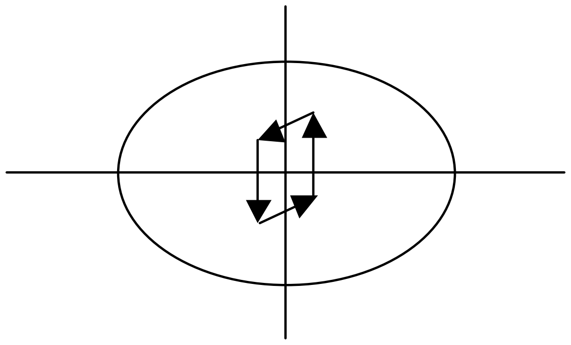


Figure 4-5: Polygon crosses international date line.

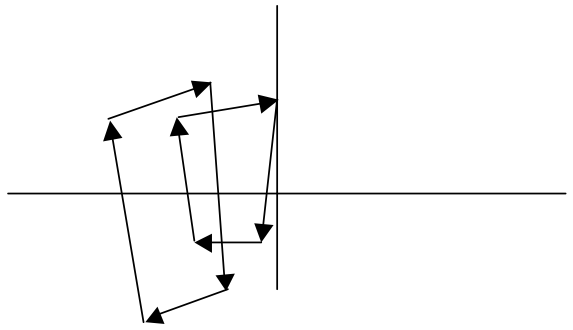
According to clock-wise order, this expression in Cartesian coordinate system represents a polygon crossing the International Date Line. Thus, it is invalid. However, in Geodetic coordinate system, this is a valid spatial coverage area. The coverage area is as shown:



4.22.4.5 Overlapping Polygons

Code Example 187: Overlapping polygons.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-20.9342</PointLongitude>
                     <PointLatitude>-11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-42.3067</PointLongitude>
                     <PointLatitude>-14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-45.7985</PointLongitude>
                     <PointLatitude>3.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-24.8982</PointLongitude>
                     <PointLatitude>6.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>-2.9342</PointLongitude>
                     <PointLatitude>-5.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-25.3067</PointLongitude>
                     <PointLatitude>-5.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-25.7985</PointLongitude>
                     <PointLatitude>3.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-0.8982</PointLongitude>
                     <PointLatitude>4.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

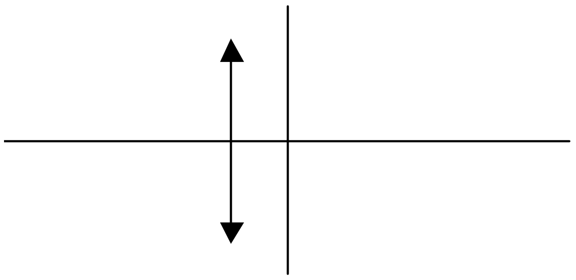


In either coordinate system, this expression represents an invalid spatial data.

4.22.4.6 Inappropriate Data

Code Example 188: Inappropriate data.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <BoundingRectangle>
         <WestBoundingCoordinate>-8.733</WestBoundingCoordinate>
         <NorthBoundingCoordinate>7.4861</NorthBoundingCoordinate>
         <EastBoundingCoordinate>-8.733</EastBoundingCoordinate>
         <SouthBoundingCoordinate>10.2617</SouthBoundingCoordinate>
      </BoundingRectangle>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

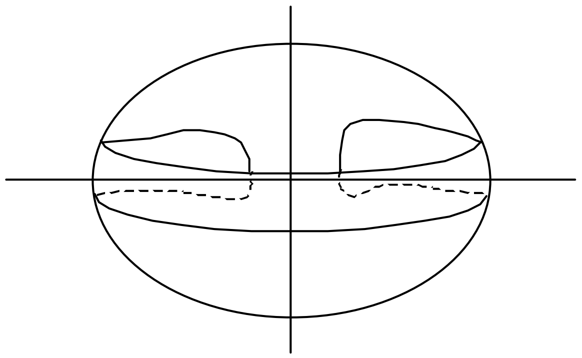


In this example, the bounding box is actually a line. It is important to use the correct spatial data type when constructing a spatial data expression. Do not use a bounding box to express a line.

> *Note: The granularity of the spatial index for a given provider can result in two distinct but nearby points being interpreted as the same point.*

4.22.4.7 Incorrect Density

If a spatial coverage area as shown below is intended in a Geodetic coordinate system:



The expression in the input file is shown as below:

Code Example 189: Example of incorrect density.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
```

```
               <Boundary>
                  <Point>
                     <PointLongitude>170.9342</PointLongitude>
                     <PointLatitude>11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-175.3067</PointLongitude>
                     <PointLatitude>14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-176.7985</PointLongitude>
                     <PointLatitude>-13.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>172.8982</PointLongitude>
                     <PointLatitude>-7.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

This expression in a Geodetic coordinate system is a valid spatial data. However, the spatial coverage area represented will be as shown below:

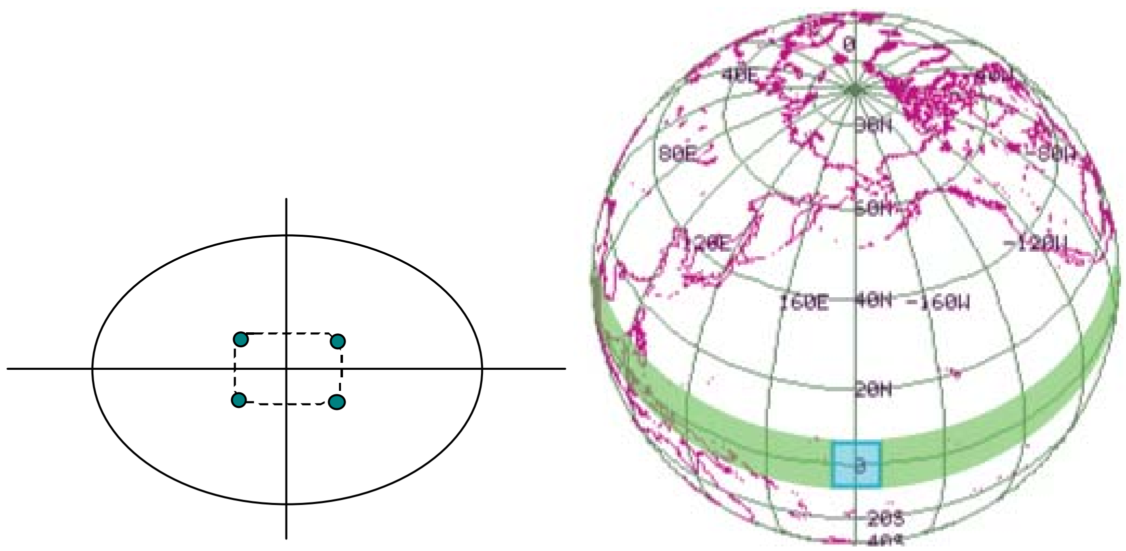


Figure 4-6: You may want the large green band, but you will get the small blue rectangle.

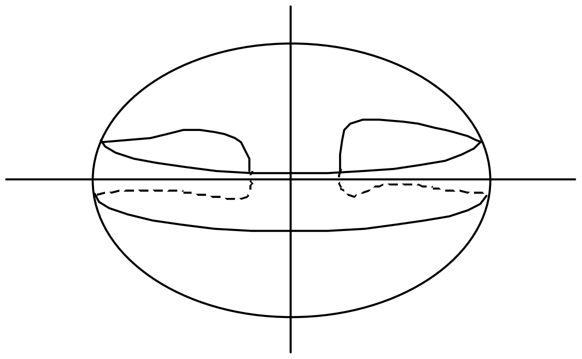
Oracle connects any two points using shortest distance between the points.

To correctly represent this spatial coverage, additional points are required to increase the density so that the spatial area can be represented. One of the expression with additional points that will represent this spatial coverage area correctly is as shown below:

Code Example 190: Using the correct density.

```
<SpatialDomainContainer>
```

```
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>170.9342</PointLongitude>
                     <PointLatitude>11.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>0.0</PointLongitude>
                     <PointLatitude>14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-175.3067</PointLongitude>
                     <PointLatitude>14.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-176.7985</PointLongitude>
                     <PointLatitude>-13.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>0.0</PointLongitude>
                     <PointLatitude>-10.1665</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>172.8982</PointLongitude>
                     <PointLatitude>-7.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```



4.22.4.8 Polygon Coverage Larger then Half of the Earth

Code Example 191: Polygon covers more than half the Earth.

```
<SpatialDomainContainer>
   <HorizontalSpatialDomainContainer>
      <Polygon>
         <SinglePolygon>
            <OutRing>
               <Boundary>
                  <Point>
                     <PointLongitude>170.9342</PointLongitude>
                     <PointLatitude>71.7045</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>0.0</PointLongitude>
                     <PointLatitude>71.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-175.3067</PointLongitude>
                     <PointLatitude>71.7732</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>-176.7985</PointLongitude>
                     <PointLatitude>-71.198</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>0.0</PointLongitude>
                     <PointLatitude>-71.1665</PointLatitude>
                  </Point>
                  <Point>
                     <PointLongitude>172.8982</PointLongitude>
                     <PointLatitude>-71.1665</PointLatitude>
                  </Point>
               </Boundary>
            </OutRing>
         </SinglePolygon>
      </Polygon>
   </HorizontalSpatialDomainContainer>
</SpatialDomainContainer>
```

The spatial area coverage in this case is much bigger then half of the earth, thus this is not a valid spatial data in a Geodetic coordinate system. This is not a valid spatial data in a Cartesian coordinate system either because the points are not expressed in clock-wise order. If the points were listed in reverse order, this spatial data would be a valid record in Cartesian coordinate system.

4.22.5 Latitude/Longitude Range and Tolerance

Tolerance and range parameter settings are used to associate a level of precision and data range with spatial data. These parameters are used as part of the evaluation parameter when validating spatial data input. For any spatial data expressed using latitude/longitude, the range is expressed in degrees and the tolerance in meters. For example, if the range for latitude is –50 to 50 and the range for longitude is –150 to 150, then any input data with latitude and/or longitude outside of this range is considered invalid. If the tolerance is 0.05 for both latitude and longitude, and if the distance between two points is less than 0.05 meter both longitudinally and latitudinally, then those two points are considered the same point. In this situation, the spatial expression is invalid because ECHO spatial constructs require each point to have unique spatial locations.

Range and tolerance parameters should be defined specifically for all data. ECHO defaults for range and tolerance are:

Latitude: -90 to 90

Longitude: -180 to 180

Tolerance for both latitude and longitude: 0.05 meters.

4.23 Browse Services

Currently, only BMGT browse image file and browse metadata processes are implemented.

4.23.1 Understanding Browse XML Input Files

For those who use BMGT to generate the metadata XML files, the generation of browse image files and browse metadata is part of the output. Both browse image files and the browse metadata XML file will be sent to the ECHO system. The ECHO system will allocate the storage for browse files, build browse image URL, and update the database to associate the browse URL to its item record.

Note: The ECHO DTD describes a mechanism for input of browse information, but browse information input in this way is not currently processed. The BMGT browse data input method should be used.

Code Example 192: Browse metadata input XML file.

```
<?xml version="1.0" encoding="UTF-8"?>
<BrowseReferenceFile>
  <DTDVersion>1.0</DTDVersion>
  <DataCenterId>EDC</DataCenterId>
  <TemporalCoverage>
    <StartDate>2002-10-17T00:00:00.000Z</StartDate>
    <EndDate>2002-10-18T00:00:00.000Z</EndDate>
  </TemporalCoverage>

  <BrowseCrossReference>
    <GranuleUR>SC:L70RWRS.002:2008440693</GranuleUR>
    <BrowseGranuleId></BrowseGranuleId>
    <InsertTime></InsertTime>
    <LastUpdate>2002-10-17 18:55:33.996</LastUpdate>
    <InternalFileName>:BR:Browse.001:2008440612:1.BINARY</InternalFileName>
    <BrowseSize>167554.0</BrowseSize>
  </BrowseCrossReference>

  <BrowseCrossReference>
    <GranuleUR>SC:L70RWRS.002:2008440702</GranuleUR>
    <BrowseGranuleId></BrowseGranuleId>
    <InsertTime></InsertTime>
    <LastUpdate>2002-10-17 18:56:04.216</LastUpdate>
    <InternalFileName>:BR:Browse.001:2008440624:1.BINARY</InternalFileName>
    <BrowseSize>159823.0</BrowseSize>
  </BrowseCrossReference>

  <BrowseCrossReference>
```

```
    <GranuleUR>SC:L70RWRS.002:2008440732</GranuleUR>
    <BrowseGranuleId></BrowseGranuleId>
    <InsertTime></InsertTime>
    <LastUpdate>2002-10-17 18:56:14.856</LastUpdate>
    <InternalFileName>:BR:Browse.001:2008440630:1.BINARY</InternalFileName>
    <BrowseSize>65258.0</BrowseSize>
  </BrowseCrossReference>
</BrowseReferenceFile>
```

The file name in the TAG "InternalFileName" is the provider's unique identifier of the browse file. ECHO supports many-to-many referencing between granules and browse files.

4.23.2 ECHO Browse Ingest Strategy

When ECHO processes the browse ingest, all the browse image files indicated by the tag <InternalFileName> under the tag of <BrowseCrossReference> in the browse metadata input XML file must be sent as well. In addition to checking the existence of the browse image files, ECHO also verifies the actual browse image file size against the file size indicated for that file in the browse metadata input XML file. If any browse image file indicated in the browse metadata input XML file does not exist, or if any file size does not match the indicated size, then the browse image files and browse metadata input XML file will not be processed.

After processing, ECHO places the browse image files online and associates them with the appropriate granule(s). This information is made available to the end user.

4.24 Sample Messages

Code Example 193: Remove all online data URLs for a granule, but not update the last update for granule Gr1 and Gr2.

```
<ProviderAccountService>
  <UpdateMetadata>
    <Granule>
      <Target>
        <ID>Gr1</ID>
        <ProviderLastUpdateDateTime>
          2002-06-03 23:00:00
        </ProviderLastUpdateDateTime>
        <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
      </Target>
      <Target>
        <ID>Gr2</ID>
        <ProviderLastUpdateDateTime>
          2002-06-03 23:00:00
        </ProviderLastUpdateDateTime>
        <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
      </Target>
      <Delete>
        <QualifiedTag>OnlineAccessURLs</QualifiedTag>
      </Delete>
    </Granule>
  </UpdateMetadata>
</ProviderAccountService>
```

Code Example 194: Deleting a specific online data URL.

```
<ProviderAccountService>
  <UpdateMetadata>
    <Granule>
      <Target>
        <ID>Gr1</ID>
        <ProviderLastUpdateDateTime>
          2002-06-03 23:00:00
        </ProviderLastUpdateDateTime>
        <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
      </Target>
      <Delete>
        <QualifiedTag>
          OnlineAccessURLs/OnlineAccessURL[URL="http://jp.provider.com/Gr1URL"]
        </QualifiedTag>
      </Delete>
    </Granule>
  </UpdateMetadata>
</ProviderAccountService>
```

Code Example 195: Insert a new online resource.

```
<ProviderAccountService>
<UpdateMetadata>
<Granule>
<Target>
  <ID>Gr1</ID>
  <ProviderLastUpdateDateTime>
    2002-06-03 23:00:00
  </ProviderLastUpdateDateTime>
  <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
</Target>
<Add>
   <QualifiedTag>GranuleOnlineResources/OnlineResource/OnlineResourceURL </QualifiedTag>
   <MetadataValue>http://jp.provider.com/Gr1URL.metadata</MetadataValue>
</Add>
<Add>
  <QualifiedTag>

GranuleOnlineResources/OnlineResource[OnlineResourceURL="http://jp.provider.com/Gr1URL.metadata"]/OnlineResourceDescription
  </QualifiedTag>
  <MetadataValue>Metadata cashed in Japan</MetadataValue>
</Add>
<Add>
  <QualifiedTag>
   GranuleOnlineResources/OnlineResource[OnlineResourceURL="http://jp.provider.com/Gr1URL.metadata"]/OnlineResourceType
  </QualifiedTag>
  <MetadataValue>Metadata</MetadataValue>
</Add>

<Add>
```

```
  <QualifiedTag>
GranuleOnlineResources/OnlineResource[OnlineResourceURL="http://jp.provider.com/Gr1URL.metadata"]/OnlineResourceMimeType
  </QualifiedTag>
  <MetadataValue>text</MetadataValue>
</Add>
</Granule>
</UpdateMetadata>
</ProviderAccountService>
```

Code Example 196: Update online data.

```
<ProviderAccountService>
<UpdateMetadata>
<Granule>
<Target>
  <ID>Gr1</ID>
  <ProviderLastUpdateDateTime>
    2002-06-03 23:00:00
  </ProviderLastUpdateDateTime>
  <SaveDateTimeFlag><DONTSAVE/></SaveDateTimeFlag>
</Target>
<Update>
  <QualifiedTag>
   OnlineAccessURL[URL="http://jp.provider.com/Gr1URL"]/URL
  </QualifiedTag>
  <MetadataValue>http://jp.provider.com/GRANULES/Gr1URL</MetadataValue>
</Update>
</Granule>
</UpdateMetadata>
</ProviderAccountService>
```

Code Example 197: QA flag update.

```
<ProviderAccountService>
<UpdateMetadata>
<Granule>
<Target>
  <ID>Gr1</ID>
  <ProviderLastUpdateDateTime>
    2002-06-03 23:00:00
  </ProviderLastUpdateDateTime>
  <SaveDateTimeFlag><SAVE/></SaveDateTimeFlag>
</Target>
<Update>
  <QualifiedTag> MeasuredParameter/MeasuredParameterContainer [ParameterName="Param1"]/QAFlags/AutomaticQualityFlag
  </QualifiedTag>
  <MetadataValue>automatic quality flag</MetadataValue>
</Update>
<Update>
  <QualifiedTag> MeasuredParameter/MeasuredParameterContainer
[ParameterName="Param1"]/QAFlags/AutomaticQualityFlagExplanation
  </QualifiedTag>
  <MetadataValue>automatic quality flag explanation</MetadataValue>
</Update>
</Granule>
</UpdateMetadata>
</ProviderAccountService>
```

4.25 Error Messages

ECHO uses different approach to process metadata updates. Metadata update is a background process. The data providers send the metadata input XML file via FTP. ECHO processes the data and sends an ingest summary report via email to the data provider upon a successful ingest. The ingest summary report is presented in XML. The status of the ingest will be reported including any abnormalities discovered in the input file. In the current version of the ingest process, the input data error handling is applied by input file pre-processing and data ingesting/updating.

4.25.1 Input File Validation Error Messages

Before the provider's data is ingested, the input file is analyzed and validated against its applied DTD. If an error is detected, the input file is rejected and an input file error report will be sent to the provider via a pre-registered email address. Below is an example of the input file error report:

Code Example 198: Error Message 1: Example of the Input File Error Report.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ECHO Ingest Input File Error Report: Wed Mar 19 10:20:01 EST 2003 -->
<!-- created  by ECHO Ingest System -->
<InputFileErrorSummary>
   <DataProvider>GCC</DataProvider>
   <ProcessStartTime>03/19/2003 10:20:01</ProcessStartTime>
   <CollectionInputFileWithError>
      <XMLValidationError>
         <FileName>VTCCLSR72001101659031903.XML</FileName>
         <ErrorMessage>
         file read took 0.005 seconds

error parsing: Error: Content model for CollectionAssociation does not allow it to end here
 in unnamed entity at line 245 char 26 of [unknown]
         </ErrorMessage>
      </XMLValidationError>
      <NoneXMLFile>VTCCLSR72001101659031903.XML.met</NoneXMLFile>
   </CollectionInputFileWithError>
   <ProcessStopTime>03/19/2003 10:20:02</ProcessStopTime>
</InputFileErrorSummary>
```

4.25.1.1 Non-XML File

If the input file does not start with the standard XML file declaration line such as <?xml …>, then this input file is considered as a "non-XML" file. A non-XML file is rejected for ingest process. The error will be reported in the input file error summary report and the error message listed as below:

```
<NonXMLFile>VTCCLSR72001101659031903.XML.met</NonXMLFile>
```

4.25.1.2 XML File Put in Wrong Entry Directory

For each provider, a directory under their FTP home directory is assigned for putting specific types of input XML files. For instance, all collection metadata XML files might go to a …/collection directory, while all granule XML files might go to a …/granule directory. If the XML file that is deposited in the …/collection directory is not a collection XML file, then this file will be rejected for ingest processing. The error will be reported in the input file error summary report and the error message listed as:

```
<None…XMLFile>VTCGMOLT200110920011240101.XML</None…XMLFile>
```

where the … is filled with specific category including "Collection", "Granule", or "Browse" depending on which category of data is examined.

4.25.1.3 Invalid XML File

Each input XML file will be validated against its corresponding DTD. If for any reason the DTD validation would fail, this file will be rejected for ingest processing. The error will be reported in the input file error summary report and the error message listed as below:

```
<XMLValidationError>
   <FileName>VTCCLSR72001101659031903.XML</FileName>
   <ErrorMessage>
      file read took 0.005 seconds
error parsing:
Error: Content model for CollectionAssociation does not allow it to end here in unnamed entity at line 245 char 26 of
[unknown]
   </ErrorMessage>
</XMLValidationError>
```

The text in the TAG of ErrorMessage is the error message generated by the XML validation utility. It usually stops validation as soon as the first error is detected.

4.25.2 Data Ingest Error Messages

When a provider's data has been successfully ingested, an ingest summary report will be sent to the provider via a pre-registered email address. Below is an example of the ingest summary report:

Error Message 23: Example of the Ingest Summary Report.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ECHO Ingest Summary Report: Wed Nov 13 13:55:01 EST 2002 -->
<!-- created  by ECHO Ingest System -->
<IngestSummary>
   <DataProvider>ETE_TEST</DataProvider>
   <IngestStartTime>11/13/2002 13:55:01</IngestStartTime>
   <Collection>
      <InputFiles>
         <File name="VTCCLSR7200211130011240555.XML" size="8424" />
      </InputFiles>
      <Processed total="1">
         <Replacement total="1">
            <DataSets>
               <DataSet shortname="L7CPF" version="2" />
            </DataSets>
         </Replacement>
      </Processed>
   </Collection>
   <Granule>
      <InputFiles>
         <File name="VTCGLSR7200211130011240101042555.XML" size="1967" />
      </InputFiles>
      <Processed total="2">
         <Insertion total="1" />
         <Replacement total="1" />
      </Processed>
   </Granule>
   <IngestStopTime>11/13/2002 13:56:26</IngestStopTime>
</IngestSummary>
```

Four groups of ingest information are included: <Collection>, <Granule>, <Browse>, <Valids>. For browse and valids, only the input files' name and size are listed to show what input files ECHO has processed. For collections and granules, more detailed information is given. The detailed ingest process information is listed under the <Processed> section of the summary report. Under the <Processed> Element, various TAG could go under each indicating a transaction or an error. The total number of collections or granules that fall into the transaction or error TAG is listed as TAG's attribute. For a collection, a list of dataset short names and version numbers will be listed for each TAG that indicates the transaction or error. ECHO breaks a large input file into small chunks to process at one time. Thus, the total in a <Process> TAG will not be bigger then 2000. The <Process> Element may be repeated multiple times for an input file.

The following list gives the possible error messages associated with each transaction within the Ingest Summary Report. Some error messages are more complex then others and so will be treated in more detail in the following subsections.

Note: A collection or granule will not be ingested if it produces any of the errors listed below.

4.25.2.1 Out of Date Update for Collection or Granule

Error Message 24: Error returned because provider's last update of collection or granule is older then the information recorded in the ECHO database.

```
<Processed total="n">
   ….
    <OutOfDate total="m">
       <DataSets>
          <DataSet shortname="collection's short name" version="version number" />
             …..
       </DataSets>
     </OutOfDate>
        …..
</Processed>
```

Where n, and m is the total occurrences for this round of ingest. If this is for granule then only total number of occurrences will be listed.

4.25.2.2 Duplicated Input

Error Message 25: Error returned due to more then one input collection or granule has the same identifier in a single round of ingest.

```
<Processed total="n">
   ….
    <Duplicated total="m">
       <DataSets>
          <DataSet shortname="collection's short name" version="version number" />
             …..
       </DataSets>
     </Duplicated>
        …..
</Processed>
```

Where n, and m is the total occurrences for this round of ingest. If this is for granule then only total number of occurrences will be listed.

This error returned because in the same round of ingest, there are more then one input collection has same identifier. ECHO reports the incident and takes the first one in the file that bearing newest collection as the one to be ingested and ignores every other ones.

4.25.2.3 Invalid Spatial Data

Error Message 26: Error returned when the spatial coverage area data for a collection or granule is invalid

```
<Processed total="n">
  ….
   <InvalidSpatial total="m">
      <DataSets>
         <DataSet shortname="collection's short name" version="version number" />
            ……
      </DataSets>
    </InvalidSpatial>
       ……
</Processed>
```

Where n, and m is the total occurrences for this round of ingest. If this is for granule then only total number of occurrences will be listed.

This error returned when the spatial coverage area data for the collection is invalid. For more description on invalid spatial data, please read section 4.22.4.

4.25.2.4 Bad Data

Error Message 27: Error returned at data loading when any piece of data associated with a collection or granule has violated a data constraint (i.e., a character string was used when a number type attribute was specified)

```
<Processed total="n">
  ….
   <RecordWithBadData total="m">
      <DataSets>
         <DataSet shortname="collection's short name" version="version number" />
            ……
      </DataSets>
    </RecordWithBadData>
       ……
</Processed>
```

Where n, and m is the total occurrences for this round of ingest. If this is for granule then only total number of occurrences will be listed.

This error is returned at data loading when any piece of data associated with a collection has violated a data constraint; such as if a character string was sent for a number type attribute. This collection will be ignored and reported back to the data provider.

However, if the data that violated the data constraint is part of the basic information of the collection, then this collection will not make it into the database. If you noticed the total number of collections processed is less then what you submitted, you should contact ECHO OPS team for an investigation. The detailed information should be in the ingest log file for OPS.

4.25.2.5 Invalid Deletion

Error Message 28: Error returned when a collection or granule identifier sent for deletion does not exist in the ECHO database

```
<Processed total="n">
   ….
    <InvalidDeletion total="m">
       <DataSets>
          <DataSet shortname="collection's short name" version="version number" />
             ……
       </DataSets>
    </InvalidDeletion>
        ……
</Processed>
```

Where n, and m is the total occurrences for this round of ingest. If this is for granule then only total number of occurrences will be listed.

4.25.2.5.1 Granule Associated with a Collection that Does Not Exist in the ECHO Database

Error Message 29: Error returned because provider's collection referenced by the granule(s) is either not in the ECHO database or failed to be ingested into the ECHO database.

```
<Processed total="n">
   ..
      <AssociatedCollectionNotExist total="m">
         ..
</Processed>
```

Where n, and m are the total occurrences for this round of ingest.

4.25.3 Errors Ignored by ECHO

In this version of ingest, ECHO do not validate the input XML file with its DTD. The incorrect input associated with the XML file is handled as discussed in this section.

Unrecognized TAG. When a TAG in the input XML file is not conformed to the DTD, the data associated with this TAG is ignored. The collection or granule will be ingested in the database with partially incorrect data (missing).

The XML input does not conform to the DTD. When an Element is placed in the XML input file without conform to the DTD, the data associated with this TAG is ignored. The collection or granule will be ingested in the database with partially incorrect data (missing).

Wrong Date Format. When the information associated with date is not expressed in a format that agreed upon between a data provider and ECHO at the time of establishment, the date information is ignored. The collection or granule will be ingested in the database with partially incorrect data (missing dates).

For instance, a data provider notified ECHO that all their date information going to be expressed in the format of "yyyy-mm-dd hh24:mi:ss" (such as "2002-03-15 14:50:00"), but the date information in the input file looks like

```
<LastUpdate>2002-3-15 2:50:00PM</LastUpdate>
```

then this date will be ignored. ECHO stores this collection or granule will empty data for the last update date.

4.26 Order Interactions With Providers

4.27 Triggers and Communication Protocols

Currently, there are three types of user actions on orders that trigger ECHO to communicate with the providers and that require the response from the providers. These three types of user actions are:

1. **SUBMIT:** This action is taken when the user is requesting to process an order. Currently, all providers must support this transaction.
2. **QUOTE:** This action is taken when the user is requesting to get a quote from a provider for an order before the order is submitted.
3. **CANCEL:** This action is taken when the user is requesting to cancel an order that is already submitted and in processing.

The provider proxy DTDs are available from the URLs in Table 4-3.

Table 4-3: Provider Proxy DTDs.

User Action	DTD
Submit order request	http://www.echo.eos.nasa.gov/dtd/v1/providerproxy/ECHOSubmitOrderRequest.dtd
Submit order response	http://www.echo.eos.nasa.gov/dtd/v1/providerproxy/ECHOSubmitOrderResponse.dtd
Cancel order request	http://www.echo.eos.nasa.gov/dtd/v1/providerproxy/ECHOCancelOrderRequest.dtd
Cancel order response	http://www.echo.eos.nasa.gov/dtd/v1/providerproxy/ECHOCancelOrderResponse.dtd
Quote order request	http://www.echo.eos.nasa.gov/dtd/v1/providerproxy/ECHOQuoteOrderRequest.dtd
Quote order response	http://www.echo.eos.nasa.gov/dtd/v1/providerproxy/ECHOQuoteOrderResponse.dtd

There are two types of protocols that ECHO communicates with the providers. The first type is solely used to communicate with an ECS-like provider. The communication uses ODL (Object Description Language) formatted messages over an open socket. The second type is to communicate using SOAP (Simple Object Access Protocol) which uses a proprietary ECHO format that is expressed in XML. These two types of communication protocols are usually referred to as either ODL (also referred to as ECS) or SOAP.

The provider must specify a URI (Uniform Resource Identifier), sometimes known as network address, in order for ECHO to communicate with the provider. The network address is usually either an IP or HTTP address. For instance, most ODL transactions point to a network location similar to the following: 64.48.91.99, while most SOAP transactions point to a network location similar to the following: http://canyon.gst.com/soap/servlet/rpcrouter. The exact URI location should be provided by the provider for each of the user actions that the provider supports. This location is where ECHO ultimately sends the actual transaction message.

4.28 Provider Response

Upon receiving any of the three order actions – submit, quote, or cancel – ECHO constructs a message with the applicable document format based on the incoming transaction and sends out the message to the specified provider location using the applicable communication protocol.

The provider must supply complete information including a ProviderTrackingID, a StatusMessage, and additional order information for transactions like submit and quote, back to ECHO.

1. **ProviderTrackingID:** A unique identifier for a provider order. ECHO provides a tracking ID on its first communication to the provider. The provider has the option of replacing it with its own tracking ID. ECHO will use the “current” tracking ID for all subsequent communications with the provider.
2. **StatusMessage:** Timestamped message records the status history of the provider order that can be viewed by the user using PresentProviderOrder or PresentOrder transaction in OrderEntryService.

3. **Additional information for submit response:** order information including total price for the order (mandatory), price for data, price for the media that stores the data, price for shipping, price for handling, any discount made to the total price, quantity of media used to store the data, estimated ship date for the order, latest data for cancellation of the order, and additional free text information.
4. **Additional information for quote response:** quote information including total price for the order (mandatory), price for data, price for the media that stores the data, price for shipping, price for handling, any discount made to the total price, recommended media to store the data, quantity of media used to store the data, expiration date for the quote (mandatory), and additional free text information.

4.28.1 Synchronous Action and Asynchronous Action

The provider is required to respond to the ECHO requested message immediately. There are two types of actions for a provider to respond to each of these user actions. The first type is synchronous action. In synchronous action, the provider sends back an answer (either accept or reject) and includes all required information such as TrackingID, StatusMessage, and additional transaction specific information in the response. The second type is asynchronous action. In asynchronous action, instead of giving an immediate answer in the response, the provider is allowed to respond with just an acknowledgement message that includes a TrackingID and a StatusMessage. To ultimately complete the request, the provider must send back its decision and any required information later via the transactions in the Provider Order Management Service.

4.28.2 Retry Mechanism

If no response is received from the provider, ECHO will issue another request after waiting for a period of time. The provider can set a policy for the maximum numbers of attempts for retry and waiting time between retries using the ProviderAccountService API.

4.28.3 Client Identity

The client identity is unique for each ECHO client. Orders created by an ECHO client carry this unique id. Providers can use the client identity to determine the source of any given order. This information is sent along with the other order information to the provider via the provider proxy.

The client identity takes the form of a string generated by the client.

Note: ECS truncates client identities at 16 characters. SOAP users have no such limitations.

This applies to the Submit, Quote, and Cancel transactions.

4.29 Data Management Service

The Data Management Service uses groups defined in the Group Management Service to grant permissions to certain users. This allows the actual membership of a group to be managed (optionally) by someone other than the data provider. This service allows a provider to deny access to everyone at one time using the aforementioned mechanisms to describe which granules and collections are affected. Then, in a subsequent transaction, permission is granted to a group or set of groups to access either all or particular granules or collections (also according to the rules).

The actual implementation of the Data Management Service centers on rules and conditions. Conditions exist independently of rules, allowing for reusability. For example, a RollingTemporalCondition of 30 days can be used in a restriction rule to prevent access to young metadata. The same RollingTemporalCondition can then be used in a permission applied to an internal test group to allow a provider to verify the correctness of said metadata. If the provider decides to change the young metadata threshold from 30 days to 45 days, only one condition must be changed (because the restriction and permission reference the same RollingTemporalCondition).

The scope of the current implementation is in terms of an access control list being applied to collections and granules as defined by the provider of the metadata. Collections can be protected such that neither the collection can be seen nor any of its constituent granules. Collections can also be protected such that the collection metadata is visible, but only certain granules are protected from view. Also, individual granules can be hidden.

The act of hiding granules or collections does not affect queries in the system. However, when the results of the query are presented, the access control lists are checked at the time of the present transaction. This means that if the access control list is updated while a user is working with a result set, the user will be able to see the things that they had been previously restricted from viewing. If an item that is hidden is presented, then an indication that it is hidden is returned instead of the actual metadata.

The Data Management Service allows the hiding of individual granules for searching and/or for ordering. It is possible to hide granules in a collection according to a rule, so the provider does not have hide each granule individually. There are three basic rules:

1. **Temporal:** Granules that are produced during a certain time period are restricted.
2. **Floating Temporal:** Granules that are produced within the last N days are restricted.
3. **Quality Flag Based:** Granules that have a certain metadata flag indicating its quality set (or cleared) are restricted.

Currently, 1 and 2 are implemented as well as the ability to identify individual granules and individual collections.

After a condition is created, a Comparator is applied to it to form a rule. Examples of Comparators include <, >, <=, =, <>, etc. Comparators and conditions form the basis for a rule's evaluation. The above mentioned 30 day RollingTemporalCondition is meaningless without the < Comparator. Similarly, a provider may create a RollingTemporalCondition of 10 years and use the > Comparator to prevent access to extremely old metadata.

The RollingTemporalCondition and TemporalCondition rules must also indicate which of three granule timestamps to apply:

1. **Production date/time:** the date and time at which the data was produced.
2. **Insert date/time:** the date and time that the metadata record for a collection or granule was inserted into the provider's metadata repository.
3. **Acquisition date/time:** the date and time that the original observation occurred.

These timestamps are referred to as the Temporal Type, an enumerated type with options PRODUCTION, INSERT, and ACQUISITION. For production and insert, standard rules apply for >=, >, <=, and <. For example, if you specify a rule using >= and the temporal type PRODUCTION, ECHO will select granules where the production_date_time timestamp is greater than or equal to the specified date.

Temporal type ACQUISITION is used in a slightly different manner. A granule's acquisition timestamp is actually represented as a range specified by a beginning acquisition date/time and an ending acquisition date/time. If you specify >= and the temporal type ACQUISITION, ECHO will select all granules where the specified date/time was greater than or equal to the beginning date/time of the granule. Choosing <= selects granules with a date/time that is equal to or earlier than the ending acquisition date/time.

Rules also accept an ActionType. Examples of ActionType include order, view, browse, etc. The ActionType allows a provider to restrict access to metadata based upon what action the user is taking. A provider may restrict the ordering of some piece of metadata, but not the browsing or discovery of the metadata.

Lastly, if the rule created is a permission, a group is associated with the rule. This allows the provider to allow access to metadata to a particular subset of the ECHO community. The group referenced by a rule does not have to include the provider as a manager. The provider creating permission may reference any group they wish.

Rules are evaluated optimistically. That is, if a particular user tries to take an action upon a piece of metadata, permissions supercede restrictions. If the metadata is restricted, but the user belongs to a group that has permission on the metadata, access is granted.

4.30 Transactions

4.30.1 ListConditions

This transaction lists the conditions the current user has created. The response contains the ConditionDescriptor, which identifies the condition, and indicates its type (temporal, rolling temporal, quality, or Boolean).

4.30.2 CreateBooleanCondition

Boolean conditions are the most straightforward of the three because they simply wrap a true/false flag.

Code Example 199: Boolean Condition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
        <CreateBooleanConditionRequest>
                <BooleanCondition>
                        <ConditionName>true condition</ConditionName>
                        <description>my true data condition</description>
                        <BooleanFlag>true</BooleanFlag>
                </BooleanCondition>
        </CreateBooleanConditionRequest>
</DataManagementService>
```

The BooleanFlag set to 'true' indicates that this is a true condition. A BooleanFlag set to 'false' would indicate a false condition. The value of the Boolean condition is evaluated at the rule level using a Comparator.

4.30.3 PresentBooleanCondition

This transaction presents the specified Boolean condition. If no names are specified, all the user's Boolean conditions are presented. The response contains a name and description of each condition, in addition to a flag that identifies the condition as Boolean.

4.30.4 CreateQualityCondition

This transaction creates a named quality condition. The QualityCondition is made up of the condition name and description, a QualityField for each condition, and a QualityThreshold for each condition. The response is an indication of whether the action succeeded.

4.30.5 PresentQualityCondition

This transaction presents a quality condition, as described above. If no names are specified, all of the user's quality conditions are presented.

4.30.6 CreateRollingTemporalCondition

Rolling temporal conditions are similar to temporal conditions in that they are both time based, however, rolling temporal conditions make use of a duration to infer a start and stop date. A provider wishing to restrict metadata produced within the previous 30 days either must modify the stop date of a temporal condition every day or use a Rolling Temporal Condition with a duration of 30 days. The rolling temporal condition is the simpler solution.

Code Example 200: Creating a rolling temporal condition with a duration of 30 days for insert date

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
   <CreateRollingTemporalConditionRequest>
      <RollingTemporalCondition>
         <ConditionName>
            young data condition
         </ConditionName>
         <description>
```

```
            young data condition
         </description>
         <Duration>
            <TemporalUnit>
               <DAYS/>
            </TemporalUnit>
            <TemporalDuration>
               30
            </TemporalDuration>
         </Duration>
         <TemporalType>
            <INSERT/>
         </TemporalType>
      </RollingTemporalCondition>
   </CreateRollingTemporalConditionRequest>
</DataManagementService>
```

You must provide a name and description for the rolling temporal condition. In addition to that information, you must also provide a temporal unit, temporal duration, and temporal type (production, acquisition, or insert – to identify the granule timestamp to use in applying the condition). These fields allow you to specify (up to the hour) what duration you are interested in controlling. The API gives you the flexibility to create a young data condition (30 Days) as well as an old data condition (10 years).

After conditions are created, rules can be enforced that reference the conditions. Each rule uses a Comparator to determine how to evaluate the embedded condition. In the Boolean and temporal condition examples, a Comparator of == makes sense (a not equals in the temporal condition would also make sense and would control all metadata NOT in the time frame specified). In the rolling temporal condition example, a Comparator of < makes sense. A > Comparator would also make sense if the rolling temporal condition had a duration of 10 years and you wished to control access to old metadata.

4.30.7 PresentRollingTemporalCondition

This transaction presents a rolling temporal condition, as described above. If no names are specified, all of the user's rolling temporal conditions are presented.

4.30.8 CreateTemporalCondition

Temporal conditions cover a specific time and are identifiable by their distinct start and stop dates. Temporal conditions are used to identify granules whose date of acquisition exists within a specific time frame. Application of a temporal condition may occur if there is a known window where a sensor on your satellite was turned off for maintenance and you wish to restrict granules during that time period.

Code Example 201: Creating a temporal condition for insert date.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
   <CreateTemporalConditionRequest>
      <TemporalCondition>
         <ConditionName>
            month of september 2001
         </ConditionName>
         <description>
            The door on the satellite was closed during september 2001.  I wish to
            block access to the metadata produced during that time period (because
            I know it to be bad).
         </description>
         <StartDate>
            <Date>
               <Month>
                  <MonthName>
                     <SEPTEMBER/>
                  </MonthName>
               </Month>
               <Day>1</Day>
               <Year> 2001</Year>
            </Date>
         </StartDate>
         <StopDate>
            <Date>
               <Month>
                  <MonthName>
                     <SEPTEMBER/>
                  </MonthName>
               </Month>
               <Day>30</Day>
               <Year>2001</Year>
            </Date>
         </StopDate>
         <TemporalType>
            <INSERT/>
         </TemporalType>
      </TemporalCondition>
   </CreateTemporalConditionRequest>
</DataManagementService>
```

When you create a temporal condition you must specify a name and a description, a start and stop date during which the temporal condition is active, and a temporal type (production, acquisition, or insert).

4.30.9 PresentTemporalCondition

This transaction presents a named temporal condition. If no names are specified, all temporal conditions will be presented. The response includes the following information:

- ConditionName
- Description
- StartTime
- StopTime
- TemporalType (production, acquisition, or insert

4.30.10 DeleteCondition

This transaction attempts to remove the specified condition. If the condition is actively enforced as a rule, the condition will not be deleted. The request must include a condition descriptor for each condition to be deleted, which identifies the condition and the condition type (temporal, rolling temporal, quality, or Boolean).

4.30.11 EnforceRule

This transaction assigns a particular data rule to the provider's data. This transaction could be used to grant visibility to a particular group for all collections for the provider. It could also be used to restrict visibility for particular collections from a provider.

The request message includes the data rule, which is comprised of:

- Rule name and description.
- Rule type (restriction or permission).
- Action type (view, order, or browse).
- Condition name.
- Condition type (temporal, rolling temporal, quality, or Boolean).

Note: Since condition names are unique for a provider, the condition type is a redundant field when submitting this data type to a transaction. However, the field is useful in responses from ECHO, because it is used to specify the exact type of Condition the rule is using. Without it, a user would be unable to present the condition.

- Comparator (less than, less than or equal, greater than, greater than or equal, equals, or not equals).
- Group name. The current implementation ignores the group name if the rule type is a restriction.

Note: The difference between a restriction and the permission is that a restriction is defined upon a user while permission is defined upon a group.

- Data holding (the collection to be restricted or allowed).

4.30.12 PresentRule

This transaction returns the details of a named data rule, as described above.

4.30.13 ListRules

This transaction generates a list of the rules the current user has assigned to their metadata holdings. The response includes a list of the current user's rules.

4.30.14 CancelRule

This transaction is not yet implemented.

4.30.15 DeleteRule

This transaction deletes a named data rule. The response indicates whether the action succeeded.

4.31 Error Messages

All errors caused by invoking transactions within the Data Management Service result in a rollback condition. That is, if you pass N rules to the EnforceRule transaction, and ECHO fails to create the Nth rule, no rules are created. This behavior is consistent with the rest of the ECHO system. The DataManagementService is only accessible to providers. Science users and guests cannot invoke any transactions in this service.

Table 4-4 gives the possible error messages associated with each transaction within the Data Management Service.

Table 4-4: DMS error messages.

Transaction	Error Message	Description
CreateBooleanCondition	<The missing item> is not valid.	The user specified a blank name or description
	Condition <name> already exists in your profile.	The user specified a name that is already in use.
CreateRollingTemporalCondition	<The missing item> is not valid.	The user specified a blank name or description
	Condition <name> already exists in your profile.	The user specified a name that is already in use.
	Invalid or null parameters were used.	The user specified a blank or non-numeric Day or Year.
	Stop date is before start date.	The user specified a stop date before the start date.
DeleteCondition	Unable to delete condition. Condition '<name>' not found.	The user specified a condition type and condition name that does not exist.
	Unable to delete condition. Condition 'null' not found.	The user specified a blank condition.
	Unable to delete condition. Condition '<name>' is actively being used in a rule.	The user specified a condition referenced by a rule.
DeleteRule	Unable to find rule: <name>.	The user specified a rule that does not exist.
	Unable to find rule: null.	The user specified a blank rule name.
EnforceRule	Unable to create rule: <the missing item> is not valid.	The user specified a blank rule name or description.
	Unable to create rule: The name '<name>' is already in use.	The user specified a rule name that already exists.
	Unable to create rule: the condition referenced does not exist.	The user specified a condition that does not exist.
	Data Value '<value>' is not valid.	The user specified a collection or granule that does not exist.
	Unable to create rule: the group specified does not exist.	The user specified a RuleType of PERMISSION but do not provide a GroupName.
PresentBooleanCondition	Boolean condition '<name>' does not exist.	The user specified a name that does not exist.
	Boolean condition 'null' does not exist.	The user specified a blank name.
PresentRollingTemporalCondition	Rolling temporal condition '<name>' does not exist.	The user specified a name that does not exist.

Transaction	Error Message	Description
	Rolling temporal condition 'null' does not exist.	The user specified a blank name.
PresentTemporalCondition	Temporal condition '<name>' does not exist.	The user specified a name that does not exist.
	Temporal condition 'null' does not exist.	The user specified a blank name.
PresentRule	Rule '<name>' does not exist.	The user specified a name that does not exist.
	Rule 'null' does not exist.	The user specified a blank name.

4.32 Creating a Restriction using a Boolean Condition

Below is an example of how a provider would restrict access to a specific collection using a Boolean Condition. After this transaction is invoked, presentation of the collection level metadata and any granule-level metadata associated with the collection will be restricted.

Code Example 202: Restricting access to a specific collection with a Boolean condition.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
   <EnforceRuleRequest>
      <DataRule>
         <RuleName>
            MODIS Level 1A V001 Restriction
         </RuleName>
         <Description>
            prevents access to the MODIS Level 1A V001 collection as well as
            all granules associated with the collection.
         </Description>
         <RuleType>
            <RESTRICTION/>
         </RuleType>
         <ActionType>
            <VIEW/>
         </ActionType>
         <ConditionType>
            <BOOLEAN/>
         </ConditionType>
         <ConditionName>
            true condition
         </ConditionName>
         <Comparator>
            <EQUALS/>
         </Comparator>
         <DataHolding>
            <DataType>
               <COLLECTION/>
            </DataType>
            <DataValue>
```

```
                MODIS Level 1A V001
            </DataValue>
        </DataHolding>
    </DataRule>
  </EnforceRuleRequest>
</DataManagementService>
```

The above example restricts viewing access on the MODIS Level 1A V001 collection using the Boolean condition named 'true'. Because the Comparator specified is EQUALS and the Boolean condition's BooleanFlag is set to 'true', access to the collection is restricted. If a provider wished to prevent viewing and ordering of the collection, they would have to create two separate restrictions and specify an ActionType of VIEW in the first restriction, and an ActionType of ORDER in the second restriction.

4.33 Restricting Access to all Metadata Holdings using a Rolling Temporal Condition

If a provider wishes to restrict access to all their metadata holdings using a Rolling Temporal Condition, they should use the DataType of COLLECTION and the DataValue of ALL (it is a reserved keyword).

Code Example 203: Rolling temporal condition restriction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
        <EnforceRuleRequest>
                <DataRule>
                        <RuleName>
                                Young Data Restriction
                        </RuleName>
                        <Description>
                                prevents access to young data
                        </Description>
                        <RuleType>
                                <RESTRICTION/>
                        </RuleType>
                        <ActionType>
                                <VIEW/>
                        </ActionType>
                        <ConditionType>
                                <ROLLING_TEMPORAL/>
                        </ConditionType>
                        <ConditionName>
                                young data condition
                        </ConditionName>
                        <Comparator>
                                <LESS_THAN/>
                        </Comparator>
                        <DataHolding>
                                <DataType>
                                        <COLLECTION/>
                                </DataType>
                                <DataValue>
                                       ALL
                                </DataValue>
                        </DataHolding>
                </DataRule>
        </EnforceRuleRequest>
</DataManagementService>
```

In the above example, the 'ALL' keyword is used to apply the rule to all collections (and their constituent granules) using a 'LESS_THAN' Comparator on a Rolling Temporal Condition with a duration of 30 days. The result of invoking this transaction is that all metadata produced within the last 30 days will be restricted from viewing.

4.34 Restrictions in ECHO Apply to all Users

Restrictions in ECHO apply to all users (including Guests). Their counterpart (permissions) apply to a group (created through the GroupManagementService) and hence affect specific users of ECHO (Guests are not included). Permissions (as their name implies) grant access to metadata that is restricted. Creating permissions is the same as creating restrictions except a Group is required when enforcing the rule.

Code Example 204: Creating permissions through conditions.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DataManagementService PUBLIC "-//ECHO DataManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/DataManagementService">
<DataManagementService>
        <EnforceRuleRequest>
                <DataRule>
                        <RuleName>
                                QA Permission
                        </RuleName>
                        <Description>
                                Allows the MODIS QA group to view all data
                        </Description>
                        <RuleType>
                                <PERMISSION/>
                        </RuleType>
                        <ActionType>
                                <VIEW/>
                        </ActionType>
                        <ConditionType>
                                <BOOLEAN/>
                        </ConditionType>
                        <ConditionName>
                                true condition
                        </ConditionName>
                        <Comparator>
                                <EQUALS/>
                        </Comparator>
                        <GroupName>
                                MODIS QA
                        </GroupName>
                        <DataHolding>
                                <DataType>
                                          <COLLECTION/>
                                </DataType>
                                <DataValue>
                                        ALL
                        </DataValue>
                    </DataHolding>
                </DataRule>
        </EnforceRuleRequest>
</DataManagementService>
```

4.35 Provider Order Management Service

4.36 Transactions

4.36.1 AcceptProviderOrderSubmission

This transaction is used when the user has submitted an order, and the provider decides to accept this order. The provider provides SubmissionInformation about this order, which includes total price, shipping/handling, latest cancel date, estimated ship date and a StatusMessage that is a short message about this order to the user. In addition, the provider is able to specify the UpdateMechanism. If the provider chooses to set this value to be "MANUAL", this means the provider is willing to update the status manually through other transactions in the Provider Order Management Service until the order is closed. In this case, the provider order state transits from SUBMITTING to PROCESSING. If the provider sets UpdateMechanism to be "NONE", this means the provider is not going to update any order status and the order will be processed in routine. In this case, the provider order state transits from SUBMITTING to CLOSED.

Note: The date format for latest cancel date and estimated ship date must follow "mm/dd/yyyy".

Code Example 205: AcceptProviderOrderSubmission transaction (updating status manually).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <AcceptProviderOrderSubmissionRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <SubmissionInformation>
         <totalPrice>10.00</totalPrice>
         <latestCancelDate>11/10/02</latestCancelDate>
         <estimatedShipDate>11/20/02</estimatedShipDate>
         <dataPrice>5.00</dataPrice>
         <mediaPrice>1.00</mediaPrice>
         <shipping>3.00</shipping>
         <handling>1.00</handling>
         <discount>0</discount>
         <quantityOfMedia>1</quantityOfMedia>
         <additionalInformation>office will close on 11/18/02</additionalInformation>
      </SubmissionInformation>
      <StatusMessage>This order has been accepted and will be processing in 24 hours.</StatusMessage>
      <UpdateMechanism>
         <MANUAL/>
      </UpdateMechanism>
   </AcceptProviderOrderSubmissionRequest>
</ProviderOrderManagementService>
```

Code Example 206: AcceptProviderOrderSubmission transaction (closing it immediately without guarantee for updating status).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <AcceptProviderOrderSubmissionRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <SubmissionInformation>
         <totalPrice>10.00</totalPrice>
         <estimatedShipDate>11/20/02</estimatedShipDate>
         <dataPrice>5.00</dataPrice>
         <mediaPrice>1.00</mediaPrice>
         <shipping>3.00</shipping>
         <handling>1.00</handling>
         <discount>0</discount>
         <quantityOfMedia>1</quantityOfMedia>
         <additionalInformation>office will close on 11/18/02</additionalInformation>
      </SubmissionInformation>
      <StatusMessage>This order has been accepted and will be shipped upon completion.</StatusMessage>
      <UpdateMechanism>
         <NONE/>
      </UpdateMechanism>
   </AcceptProviderOrderSubmissionRequest>
</ProviderOrderManagementService>
```

4.36.2 RejectProviderOrderSubmission

This transaction is used when the user has submitted the order, and the provider decides to reject this order. The provider provides the user a StatusMessage to explain the reason of rejection. In this transaction, the provider order state transits from SUBMITTING to SUBMIT_REJECTED.

Code Example 207: RejectProviderOrderSubmission transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <RejectProviderOrderSubmissionRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>You are not allowed to order item #111 because … </StatusMessage>
   </RejectProviderOrderSubmissionRequest>
</ProviderOrderManagementService>
```

4.36.3 CancelProviderOrder

This transaction is used either when the user has requested to cancel the provider order using the CancelOrder transaction in the User Account Service and the provider decides to accept the request or when the provider decides to cancel the provider order due to whatever the reason is. In this first case, the provider order state transits from CANCELLING to CANCELLED. In this second case, the provider order state transits from PROCESSING to CANCELLED. In any case, the provider must provide a StatusMessage about this cancellation.

Code Example 208: CancelProviderOrder transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <CancelProviderOrderRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>This order is cancelled as requested. </StatusMessage>
   </CancelProviderOrderRequest>
</ProviderOrderManagementService>
```

4.36.4 RejectProviderOrderCancellation

This transaction is used when the user has requested to cancel the provider order using the CancelOrder transaction in the User Account Service and the provider decides to reject this request. The provider must provider a StatusMessage explaining the reason of rejection. In this transaction, the provider order state transits from CANCELLING to PROCESSING.

Code Example 209: RejectProviderOrderCancellation Transaction

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <RejectProviderOrderCancellationRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>The item is ready to ship out, can not cancel anymore. </StatusMessage>
   </RejectProviderOrderCancellationRequest>
</ProviderOrderManagementService>
```

4.36.5 CloseProviderOrder

This transaction is used when the provider has finished processing at least part of the order. The provider will provide the user a StatusMessage about the progress of the order. In this transaction, the provider order state transits from PROCESSING to CLOSED.

Code Example 210: CloseProviderOrder transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <CloseProviderOrderRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>Your order has been shipped out.</StatusMessage>
   </CloseProviderOrderRequest>
</ProviderOrderManagementService>
```

4.36.6 SupplyProviderQuote

This transaction is used when the user has requested to quote the provider order using the QuoteOrder transaction in the Order Entry Service and the provider decides to supply the information quoted. The provider provides a

ProviderQuote that includes the total price, shipping/handling, recommended media, quote expiration date etc. and a StatusMessage to the user. In this transaction, the provider order state transits from QUOTING to QUOTED. Note that the date format for quote expiration date must follow “mm/dd/yyyy”.

Code Example 211: SupplyProviderQuote transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <SupplyProviderQuoteRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <ProviderQuote>
         <totalPrice>8.00</totalPrice>
         <quoteExpirationDate>11/30/02</quoteExpirationDate>
         <dataPrice>5.00</dataPrice>
         <mediaPrice>1.00</mediaPrice>
         <shipping>3.00</shipping>
         <handling>1.00</handling>
         <discount>0.20</discount>
         <quantityOfMedia>1</quantityOfMedia>
         <recommendedMedia>CD</recommendedMedia>
         <additionalInformation>office will close on 11/18/02</additionalInformation>
      </ProviderQuote>
      <StatusMessage>quote as requested. </StatusMessage>
   </SupplyProviderQuoteRequest>
</ProviderOrderManagementService>
```

4.36.7 RejectProviderQuote.

This transaction is used when the user has requested to quote the provider order using the QuoteOrder transaction in the Order Entry Service and the provider decides to reject the quote request. The provider provides the user a StatusMessage to explain the reason of rejection. In this transaction, the provider order state transits from QUOTING to QUOTE_REJECTED.

Code Example 212: RejectProviderQuote transaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagerService PUBLIC "-//ECHO ProviderOrderManagementService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/ProviderOrderManagementService">
<ProviderOrderManagementService>
   <RejectProviderQuoteRequest>
      <ProviderTrackingID>123456789</ProviderTrackingID>
      <StatusMessage>Our system does not support quote at this time. </StatusMessage>
   </RejectProviderQuoteRequest>
</ProviderOrderManagementService>
```

4.36.8 UpdateStatusMessage.

This transaction allows the provider to update the status message of an order to inform the user. This transaction does not associate with any provider order state change.

4.36.9 ChangeTrackingID

The tracking ID is the identity of ECHO order in the provider's system. In some cases, the provider wants to change this ID in their system. This transaction is used to register this change to ECHO system.

4.36.10 Provider Order History

Each provider has the capability to present the history of the orders that have been submitted to its data center. The PresentClosedOrder transaction lists all the orders that have been closed and the PresentOpenOrder lists all the orders that have been submitted to the provider but are still in process. Each of these transactions lists detailed information of each order. To list only the order Id and its order state, POMS provides PresentClosedOrderSummary and PresentOpenOrderSummary.

4.36.10.1 PresentClosedOrder

This transaction enables a provider to view detailed information about the orders that are no longer active. The provider has the choice of either specifying or not specifying the provider order states in the request. If provider order states are specified, the information of orders in the specified state is returned. If no provider order state is specified, then all the persisted orders that are closed are returned.

Code Example 213: Present all closed orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService PUBLIC "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentClosedOrderRequest/>
</ProviderOrderManagementService>
```

Code Example 214: Present closed orders in CLOSED state.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService PUBLIC "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentClosedOrderRequest>
      <ClosedProviderOrderState>
         <CLOSED/>
      </ClosedProviderOrderState>
   </PresentClosedOrderRequest>
</ProviderOrderManagementService>
```

4.36.10.2 PresentClosedOrderSummary

This transaction enables a provider to view brief information about the orders that are no longer active. The provider has the choice of either specifying or not specifying the provider order states in the request. If provider order states are specified, the information of orders in the specified state is returned. If no provider order state is specified, then all the persisted orders that are closed are returned.

Code Example 215: Present summary for all closed orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService PUBLIC "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentClosedOrderSummaryRequest/>
```

```
</ProviderOrderManagementService>
```

Code Example 216: Present summary for closed orders in CLOSED state.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService PUBLIC "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentClosedOrderSummaryRequest>
      <ClosedProviderOrderState>
         <CLOSED/>
      </ClosedProviderOrderState>
   </PresentClosedOrderSummaryRequest>
</ProviderOrderManagementService>
```

4.36.10.3 PresentOpenOrder

This transaction enables a provider to view detailed information about the orders that are still open. The provider has the choice of either specifying or not specifying the provider order states in the request. If provider order states are specified, the information of orders in the specified state is returned. If no provider order state is specified, then all the persisted orders that are still open are returned.

Code Example 217: Present all open orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService PUBLIC "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentOpenOrderRequest/>
</ProviderOrderManagementService>
```

Code Example 218: Present open orders in PROCESSING state.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService PUBLIC "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentOpenOrderRequest>
      <OpenProviderOrderState>
         <PROCESSING/>
      </OpenProviderOrderState>
   </PresentOpenOrderRequest>
</ProviderOrderManagementService>
```

4.36.10.4 PresentOpenOrderSummary

This transaction enables a provider to view brief information about the orders that are still open. The provider has the choice of either specifying or not specifying the provider order states in the request. If provider order states are specified, the information of orders in the specified state is returned. If no provider order state is specified, then all the persisted orders that are still open are returned.

Code Example 219: Present summary for all open orders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService PUBLIC "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentOpenOrderSummaryRequest/>
</ProviderOrderManagementService>
```

Code Example 220: Present summary for open orders in PROCESSING state.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ProviderOrderManagementService PUBLIC "http://localhost:7001/echo/dtd/ProviderOrderManagementService.dtd">
<ProviderOrderManagementService>
   <PresentOpenOrderSummaryRequest>
      <OpenProviderOrderState>
         <PROCESSING/>
      </OpenProviderOrderState>
   </PresentOpenOrderSummaryRequest>
</ProviderOrderManagementService>
```

4.37 Provider Order Management Service Error Messages

Table 4-5 gives the possible error messages associated with each transaction within the Provider Order Management Service.

Table 4-5: POMS error messages.

Transaction	Error Message	Description
AcceptProviderOrderSubmission RejectProviderOrderSubmission CancelProviderOrder RejectProviderOrderCancellation SupplyProviderQuote RejectProviderQuote CloseProviderOrder UpdateStatusMessage ChangeTrackingID PresentOpenOrder PresentOpenOrderSummary PresentClosedOrder PresentClosedOrderSummary	[SessionManager] The service {ProviderOrderManagementService} that you attempted to access either does not exist or you do not have the appropriate access level to use it.	This error is returned when trying to access this transaction by a user who does not have the privilege for this transaction.
	Order <OrderID> does not exist in the system	This error is returned when trying to access an order using the order ID that does not exist for this user.
	Invalid provider tracking Id.	This error is returned when trying to access an order using the tracking ID that does not exist for this provider.
	Invalid state for ProviderOrder.	This error is returned when trying to update order from one state to the other state that does not conform to the pre-defined order state flow (e.g., A provider try to set the order state to be “RejectSubmission” on an order at the state of “Closed”).
	[SessionManager] The service {ProviderOrderManagementService} that you attempted to access either does not exist or you do not have the appropriate access level to use it.	This error is returned when trying to access this transaction by a user who does not have the privilege for this transaction.

5 DEVELOPERS NOTES

5.1 Leverage Address Book and Preferences

A good feature in the ECHO system is that a registered user is able to keep an address book. In the address book, a user can assign a meaningful name for each of the addresses, for example, a home address, a working address, and a project address, etc. The address book could be convenient for the user when an address is to be filled in, for instance filling in the contact, shipping, and billing address when placing an order. Instead of filling in each address line by line, the user can simply set up the preferences on which name of the address in the address book is to be filled in for the contact, shipping, and billing address respectively. In this way, the client developer can easily write a program to retrieve the address from the address book and automatically set the contact, shipping and billing address according to the user's preferences. The following is an example in greater detail showing the step-by-step procedures for a client developer to set the contact, shipping, and billing address for a registered ECHO user.

The first step is to retrieve the user's preferences on the contact, shipping, and billing address. This can be achieved by using the PresentOptionSelectionsForUser transaction in the UserAccountService. Code Example to present preferences for contact, shipping, and billing address is below.

Code Example 221: Requesting message.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentOptionSelectionsForUserRequest>
      <OptionName>default_contact_address</OptionName>
      <OptionName>default_shipping_address</OptionName>
      <OptionName>default_billing_address</OptionName>
   </PresentOptionSelectionsForUserRequest>
</UserAccountService>
```

Code Example 222: Response message.

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
  <PresentOptionSelectionsForUserResponse>
    <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED />
      </BooleanResultType>
    </BooleanResult>
    <OptionSelection>
      <SimpleOptionSelection>
       <OptionName>default_contact_address</OptionName>
       <Value>home</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_shipping_address</OptionName>
        <Value>project</Value>
      </SimpleOptionSelection>
    </OptionSelection>
    <OptionSelection>
      <SimpleOptionSelection>
        <OptionName>default_billing_address</OptionName>
        <Value>work</Value>
      </SimpleOptionSelection>
    </OptionSelection>
  </PresentOptionSelectionsForUserResponse>
</UserAccountService>
```

If the user has not yet set all of the preferences on the contract, shipping, and billing address, the client developer can let the user either fill each address fields directly, or give the user a chance to set the preferences. To set the preferences, use the SetOptionSelectionsForUser transaction in the UserAccountService.

Code Example 223: Set option selections for user.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <SetOptionSelectionsForUserRequest>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_shipping_address</OptionName>
            <Value>project</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_billing_address</OptionName>
            <Value>work</Value>
         </SimpleOptionSelection>
      </OptionSelection>
      <OptionSelection>
         <SimpleOptionSelection>
            <OptionName>default_contact_address</OptionName>
            <Value>home</Value>
         </SimpleOptionSelection>
      </OptionSelection>
   </SetOptionSelectionsForUserRequest>
</UserAccountService>
```

The second step is to retrieve the address from the user's address book according to the user's preferences. To serve the purpose, use the PresentAddressInformation transaction in the UserAccountService. Code example for presenting an address named "work" is as below:

Code Example 224: Request message to present an address called "work."

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
   <PresentAddressInformationRequest>
      <AddressID>work</AddressID>
```

```
  </PresentAddressInformationRequest>
</UserAccountService>
```

Code Example 225: Response message to present an address called "work."

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE UserAccountService PUBLIC "-//ECHO UserAccountService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/UserAccountService.dtd">
<UserAccountService>
  <PresentAddressInformationResponse>
    <BooleanResult>
      <BooleanResultType>
        <REQUEST_SUCCEEDED />
      </BooleanResultType>
    </BooleanResult>
    <AddressInformation>
      <AddressID>work</AddressID>
      <USFormat>True</USFormat>
      <Street1>7855 Walker Drive</Street1>
      <Street2>Suite 200</Street2>
      <City>Greenbelt</City>
      <State>MD</State>
      <Zip>20770</Zip>
      <Country>USA</Country>
    </AddressInformation>
  </PresentAddressInformationResponse>
</UserAccountService>
```

The last step is to load the address fields retrieved from the user's address book to the place the address needs to be filled in. For example, a billing address is to be filled in. From the first example, we know the user's preferred billing address is "work." In the second step, we retrieve the address "work" from the user's address book as seen in the PresentAddressInformation response message. In this last step, the client developer parses all the address fields from the response message and fill in the corresponding address fields in the billing address for the user automatically.

5.2 Present Results Strategy

The ECHO system provides sufficient flexibility in presenting query results to help client developers build their customized user-friendly interface. Below are some tips for making the best use of the ECHO system.

5.3 Present Minimum Interested Attributes

ECHO allows results to be presented according to specified TupleType in the PresentationDescription. This is a very useful feature because different group of users may have limited and different interests in what should be presented. For example, some groups may only have interest in the "Instrument" attribute of the result. They can specify "Instrument" in the TupleType when issuing a present request. The present response thus only includes the "Instrument" related information. In this way, users can get exactly what they want to see. Meanwhile, this can also reduce the response time when communicating with ECHO, because otherwise every attribute of the query results is returned if no TupleType is specified, making it a time-consuming process.

Code Example 226: Presenting a result with TupleTypes GranuleUR, ShortName, and Instrument.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
```

```
<CatalogService>
   <PresentRequest>
      <ResultSetID>RU10021060199867864</ResultSetID>
      <PresentationDescription>
         <TupleType>
            <attributeName>GranuleUR</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>ShortName</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <TupleType>
            <attributeName>Instrument</attributeName>
            <PrimitiveTypeName>
               <String/>
            </PrimitiveTypeName>
         </TupleType>
         <DTDType>
            <ECHO/>
         </DTDType>
      </PresentationDescription>
      <IteratorSize>5</IteratorSize>
      <Cursor>1</Cursor>
   </PresentRequest>
</CatalogService>
```

In the presentation response, other than the specified the attributes, the parent nodes all the way to the root node as well as any required nodes are presented. This is because of the tree structure of XML document and that the XML document must conform to its DTD. The child node information is not included or is minimum. However, users may have equal or more interest looking at the information in the child nodes under the specified attributes. If this is the case, the child attributes must be also specified. For example, a user group is interested in "Instrument" information, but may as well want to look at "Sensor", a node under "Instrument". It is good for the client developer to let the users familiar to the tree structure in order to meet users' true need with efficiency.

5.3.1 Use Paging Mechanism

A good user interface for displaying query results should include a fixed and small number of results at each page, and allow users to scroll down to the next pages for more results. The results in each page should be hyper-linked with short descriptions. Users can hit each hyperlink and get the detailed information for that result.

ECHO allows Clients to set the Cursor and IteratorSize in presenting query results, which that makes paging easily applied. For example, a client developer wants each page displaying 10 results. Displaying results in the first page, the present request message should set `<IteratorSize>10</IteratorSize>` and `<Cursor>1</Cursor>`; for the second page, the request message should set `<IteratorSize>10</IteratorSize>` and `<Cursor>11</Cursor>`, and so on for the rest of the result pages.

For better performance, client developers can issue a present request for each result page in a "pre-fetch" manner. In other words, present requests are performed for every page even before the user clicks a button asking for a particular page. Immediate results are received without any delay in communicating with the ECHO server, and may win more satisfied end-users.

5.4 XML Development Tools

ECHO is an XML message based system. During development of the ECHO system, many XML tools are used to aid the design and development. The sections below introduce the XML tools used by the ECHO development team.

5.5 XML Spy

XML Spy is an editing and validation tool for XML files. XML Journal, Java Journal, and PC Magazine voted XML Spy as the best XML editing and evaluation tool currently available. The following are just some of the features offered by XML Spy:

- XML editing and validation
- Schema and DTD design
- XSL transformation.

For more detailed information, please view http://link.xmlspy.com/.

5.6 SAX Parser and DOM Parser

ECHO employs Apache's XML parser Xerces and XSLT's stylesheet processor xalan for XML file and message processing.

5.6.1 Xerces

The statement below is quoted from the Apache website on Xerces:

> "Xerces (named after the Xerces Blue butterfly) provides world-class XML parsing and generation. Fully-validating parsers are available for both Java and C++, implementing the W3C XML and DOM (Level 1 and 2) standards, as well as the de facto SAX (version 2) standard. The parsers are highly modular and configurable. Initial support for XML Schema (draft W3C standard) is also provided. "

A Perl wrapper is provided for the C++ version of Xerces, which allows access to a fully validating DOM XML parser from Perl. It also provides for full access to Unicode strings, since Unicode is a key part of the XML standard. A COM wrapper (also for Xerces-C) provides compatibility with the Microsoft MSXML parser. ECHO uses both SAX parser and DOM parser for java provided by Xerces.

For more information about Apache Xerces, please view http://xml.apache.org/xerces2-j/index.html.

5.6.2 Xalan

The statement below is quoted from the Apache website on xalan:

"Xalan (named after a rare musical instrument) provides high-performance XSLT stylesheet processing. Xalan fully implements the W3C XSLT and XPath recommendations. The stylesheet processor is feature-rich and robust. The XPath Processor is useable as a stand-alone unit. Xalan uses the Bean Scripting Framework (BSF) to implement Java and script extensions, features EXSLT extensions, nodeset extension, multiple document output extensions and SQL extension. "

Xalan is currently available in Java, and C++. ECHO uses Xalan for java to perform XML file/message conversion.

For more information on Apache's Xalan, please view http://xml.apache.org/xalan-j/index.html.

6 ECHO PROVIDER USER'S MANAGEMENT PROGRAM (PUMP)

6.1 OVERVIEW

PUMP (Provider User Management Program) is a User Interface (UI) that helps Earth Observing System (EOS) Clearing House (ECHO) data providers easily manage their data and services entered as part of and provided by ECHO.

This document provides basic information on the use of the PUMP User Interface. It describes what PUMP is, where to get it, and how to use it.

6.1.1 Background (How PUMP relates to ECHO)

ECHO is an enabling framework that allows different data systems and services to work together. ECHO helps data providers make their Earth science resources available to the science community.

An ECHO data provider is an entity that participates with ECHO to provide Earth science data in the form of its metadata holdings. This metadata is available for search and retrieval as well as navigation and discovery. Providers have complete control over how metadata is represented in ECHO on their behalf, as well as who has access to their data.

There are three types of registered users: Science Users, Providers, and Administrative Users. An Administrative User is responsible for administering the ECHO system, including data and other users of ECHO. The Administrative User has some privileges such as being able to login as any other user in the system, delete users, grant special privileges to users, and revoke special privileges to users. A Science User is any person who wishes to use ECHO to search for and order Earth science data and services. Some Science Users have a special privilege, the ability to act on behalf of a Provider. It is these users that would use the PUMP tool.

A Provider is an organization with products and/or services that are searchable/orderable through ECHO. Earth Scientists search and order Providers' data and services.

In PUMP, a provider is closely linked to a specific registered user because to perform provider activities, a registered user needs to work on behalf of that provider. PUMP is a UI that allows registered users acting on behalf of a provider perform administrative or data management activities in ECHO.

6.1.2 Description of PUMP UI

The PUMP UI or PUMP client consists of a set of different Graphical User Interface (GUI) screens. Each screen is used to perform different processes to control access to metadata.

Most of the work done with PUMP is done in the working screens. The working screen layout is mainly divided in three parts: the ECHO server session area, the navigator, and the data display/work area. Figure 6-1 shows a working screen:

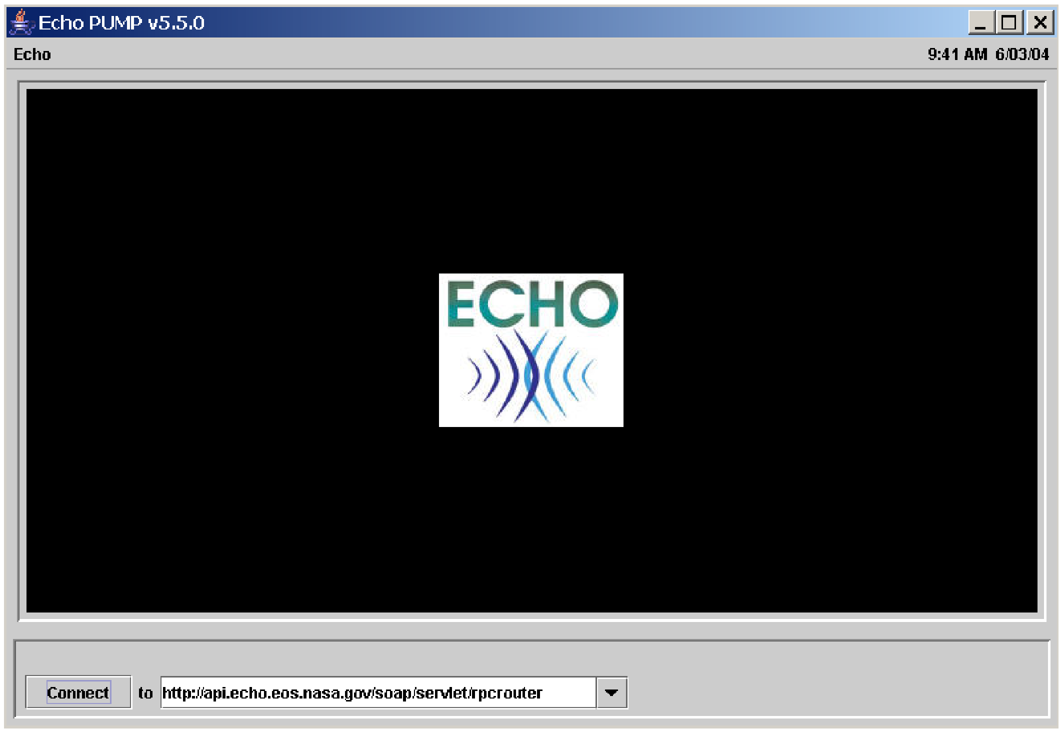


Figure 6-1. PUMP initial screen.

1. The ECHO server session area is located at the bottom of the screen. It allows you to set/leave context to a different user, set/leave context to a provider, login/logout as a user, and connect/disconnect to the ECHO server.
2. The Navigator part of PUMP is a list of radio buttons that appear on the left side of the screen. Selecting the different buttons allows you to navigate through the system so you can perform the different services providers need to do to manage their data.
3. The data display/work area consumes most of the screen; it is in the right upper area. This section works with the navigator. When you select a button from the navigator, this area displays the work area for the selected service.

6.2 HOW TO USE PUMP

To use PUMP, you need to connect to the appropriate echo server where you need to have access to your metadata. Since PUMP is a User Interface used by ECHO providers, you should have an account so you can login to it. See section 6.2.2 below to register to ECHO and get an account.

Once you are logged in to PUMP, you will be presented with the working screens that allow you to manage the data. The sequence of events to start using PUMP is shown with Figure 6-2:

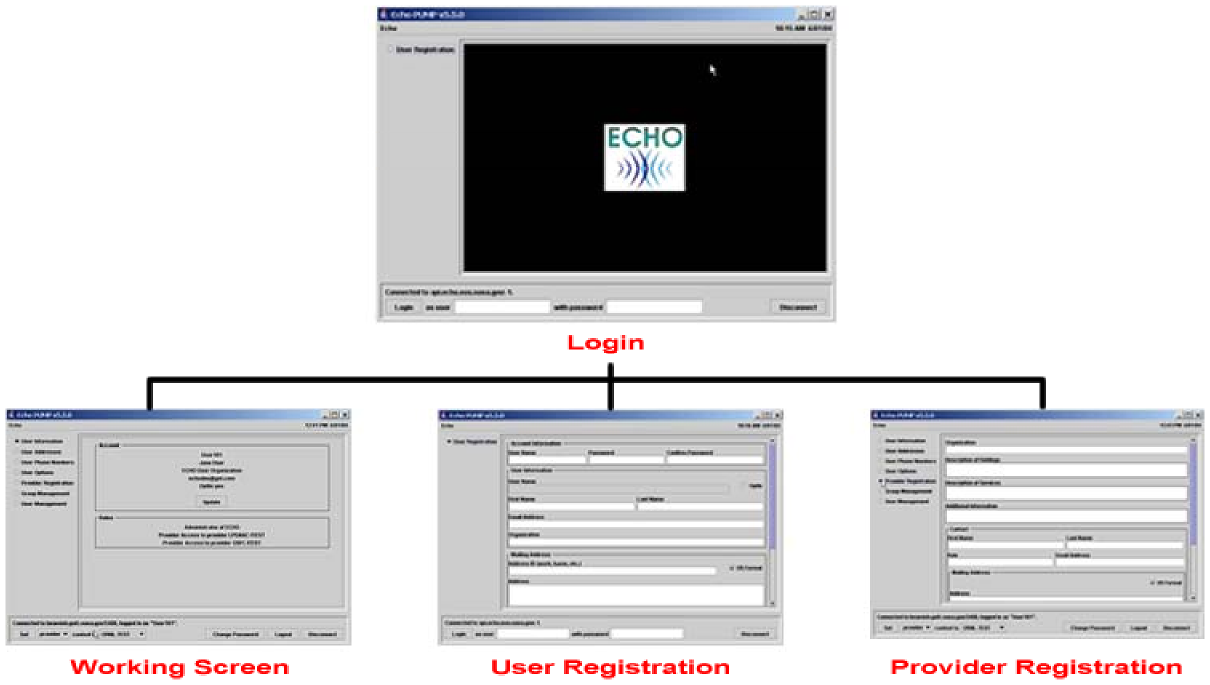


Figure 6-2. Using PUMP.

6.2.1 How to get and install PUMP

1. Go to the following link:

   http://www.echo.eos.nasa.gov/pump_releases/V_5.5.0/dist/webstart/pump.jnlp.

2. Download the version that is appropriate for your system.

3. Follow the installation instructions on the page.

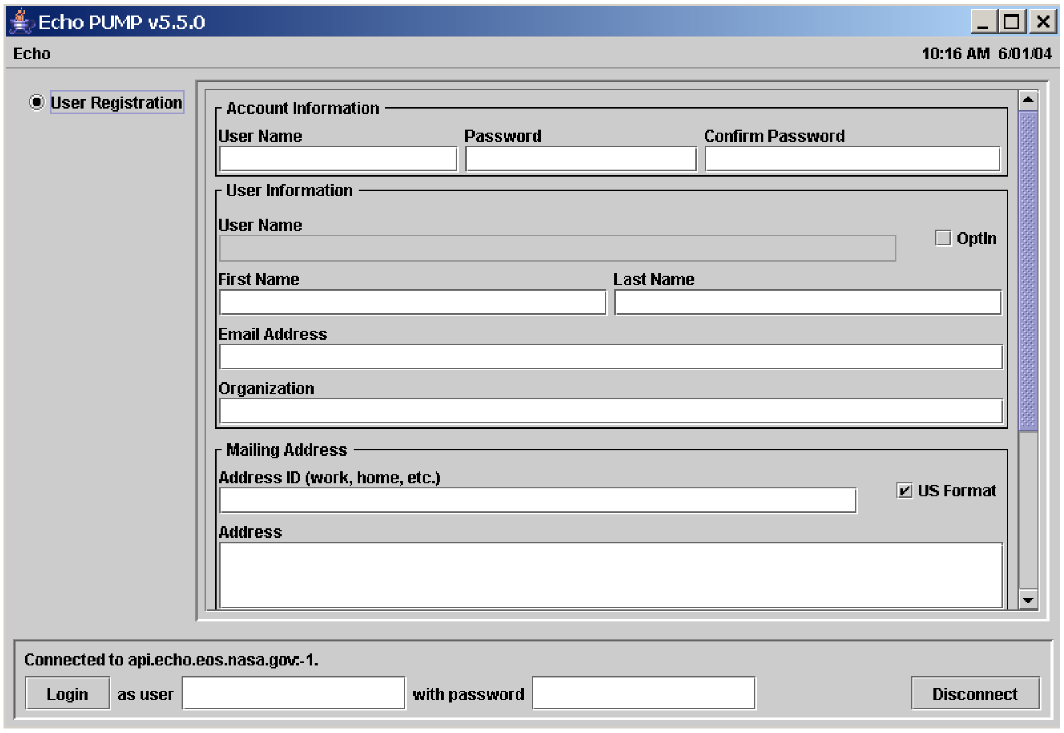
6.2.2 Registration Service

The Registration Service is the interface that allows a user to become a registered user. It also allows potential providers to submit the provider application.

6.2.2.1 Becoming a Registered User

To become a registered user using PUMP:

1. From the login page (Fig. 6-3), select the "User Registration" button on the navigator, upper left area of the screen. The User Registration screen (Figure 6-3) will appear.
2. Fill in all the input fields and click the "Register" button. You will get a message indicating successful registration.



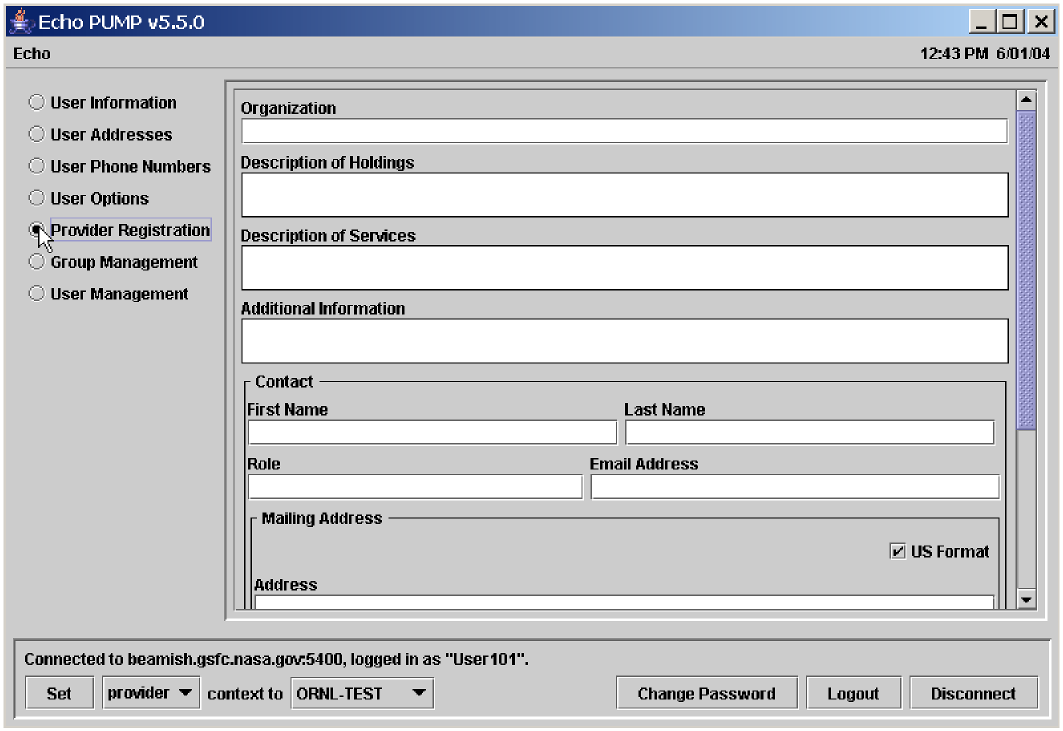
Connected to api.echo.eos.nasa.gov:-1.

Figure 6-3. Registering as a user.

6.2.2.2 Becoming a Provider

To submit an application to become a provider of data or services using PUMP, from the login page (Fig. 6-6):

1. After creating a registered account (6.2.2.1), Login into the newly created account.
2. Select the "Provider Registration" button on the navigator, upper left part of the screen. The "Provider Registration" screen (Figure 6-4) will appear.
3. Fill in all the input fields and click the "Register" button. You will get a message indicating successful registration and a provider ID number. Be sure to record your provider ID number.



Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101".

Figure 6-4. Registering as a provider.

6.2.3 Connecting to ECHO

To connect to the ECHO server using PUMP:

1. When you launch the PUMP UI you will get the connection screen (Figure 6-5):
2. Either select the needed ECHO server from the pull down selection list, or type the ECHO server location into the input field. Click on the "Connect" button. The login screen (Fig.6-6) will appear.
3. To cancel your connection to the ECHO server, click on the "Disconnect" button at the bottom of the screen.

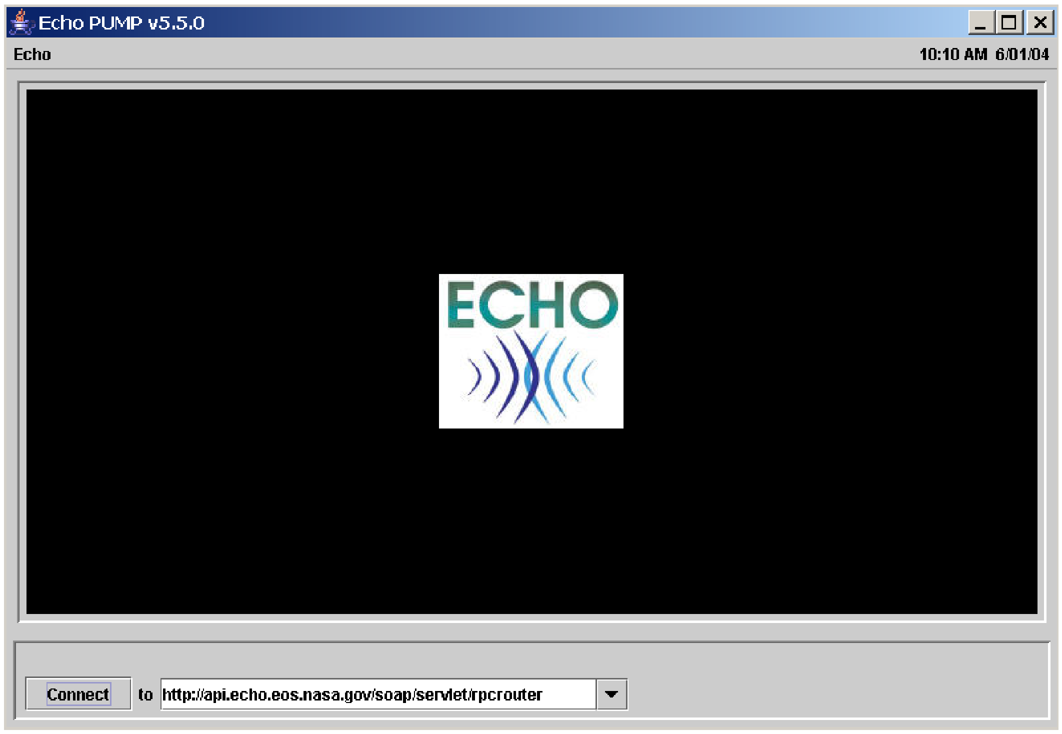


Figure 6-5. Connection screen.

6.2.4 Logging in to ECHO

Once you are connected to the ECHO server, you will see the login screen (Figure 6-6).

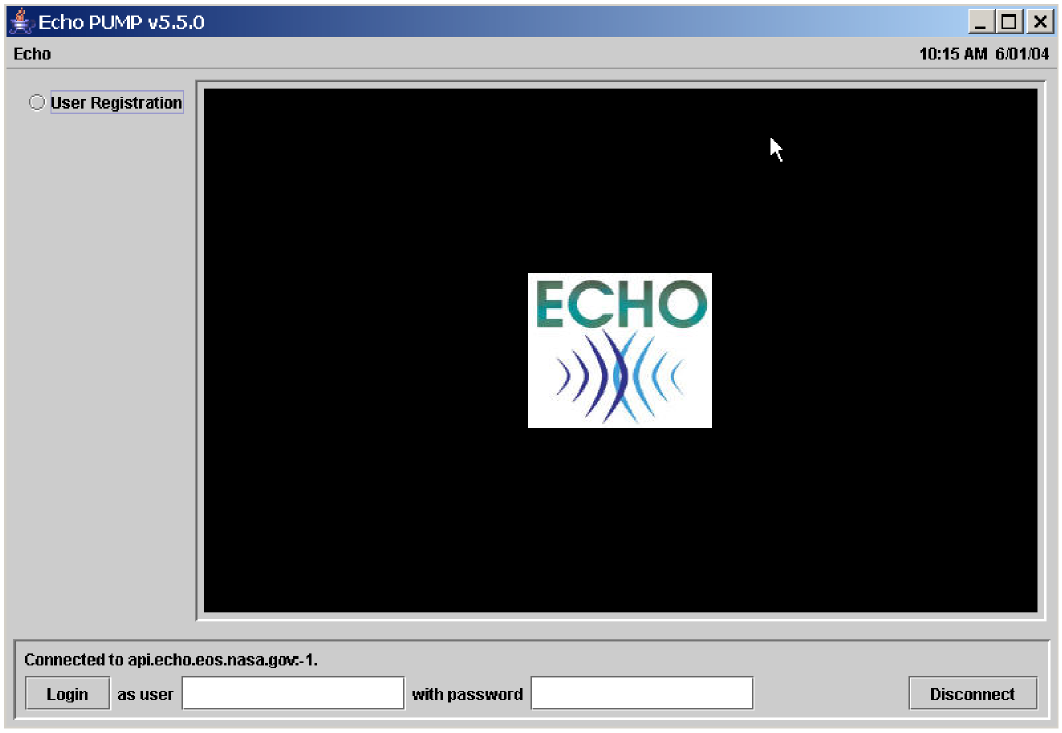
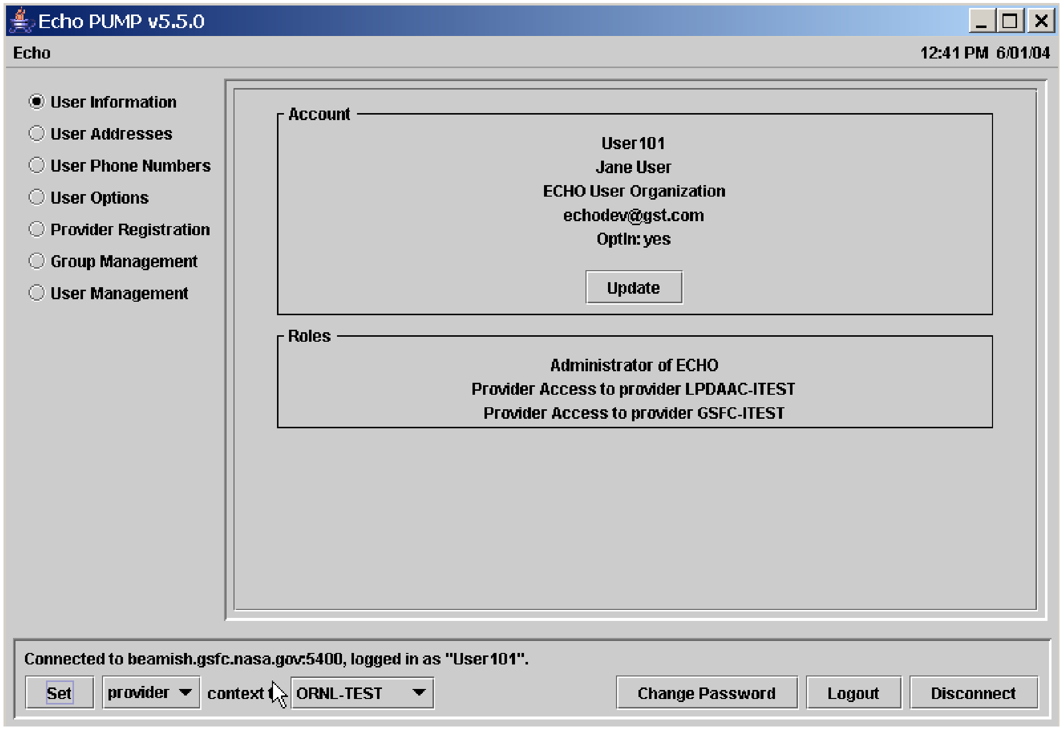


Figure 6-6. Login screen.

1. Type your user name and your password in the input fields and click the login button. If you don't have an ECHO account, see the "Registration Service" section.
2. Once you are logged in to the ECHO server via PUMP, you will be presented with the User Information screen (Figure 6-7).
3. From this point on you will be allowed to perform different activities depending on the level of permissions you have. The permissions are based on the type of user you are (Administrative or Provider). See "Registration Service" (section 6. 2.2) for details on becoming an ECHO user.



User101
Jane User
ECHO User Organization
echodev@gst.com
OptIn: yes

Administrator of ECHO
Provider Access to provider LPDAAC-ITEST
Provider Access to provider GSFC-ITEST

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101".

Figure 6-7. User information screen.

6.2.5 Setting User Context

Administrative users are allowed special transactions for ensuring the integrity of ECHO. One transaction that administrative users can do is to "become" any other user in the system. If you are an administrative user and need to act as another user, you must change the user context to that user. Since administrative users have the function to grant and/or revoke privileges to users, a way for an administrative user to verify that the change in privilege is working is by acting as that user and seeing that the changes in privileges took effect.

To change the user context to another user:

1. From the server session area located at the bottom of the screen, select the "user" option from the first pull-down menu; enter the user id in the input field (Figure 6-8).
2. Click the "Set" button on the left. The status bar (Figure 6-9) will update itself.
3. To leave the user context, just click the "Leave User Context" button.

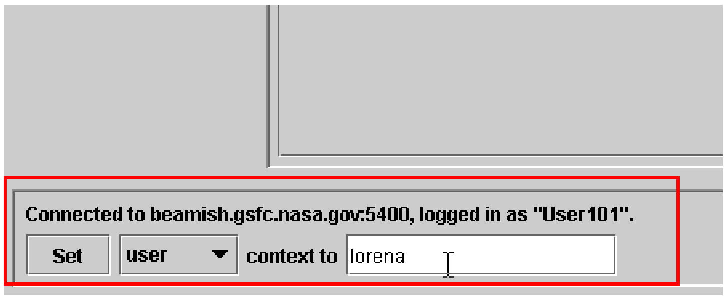
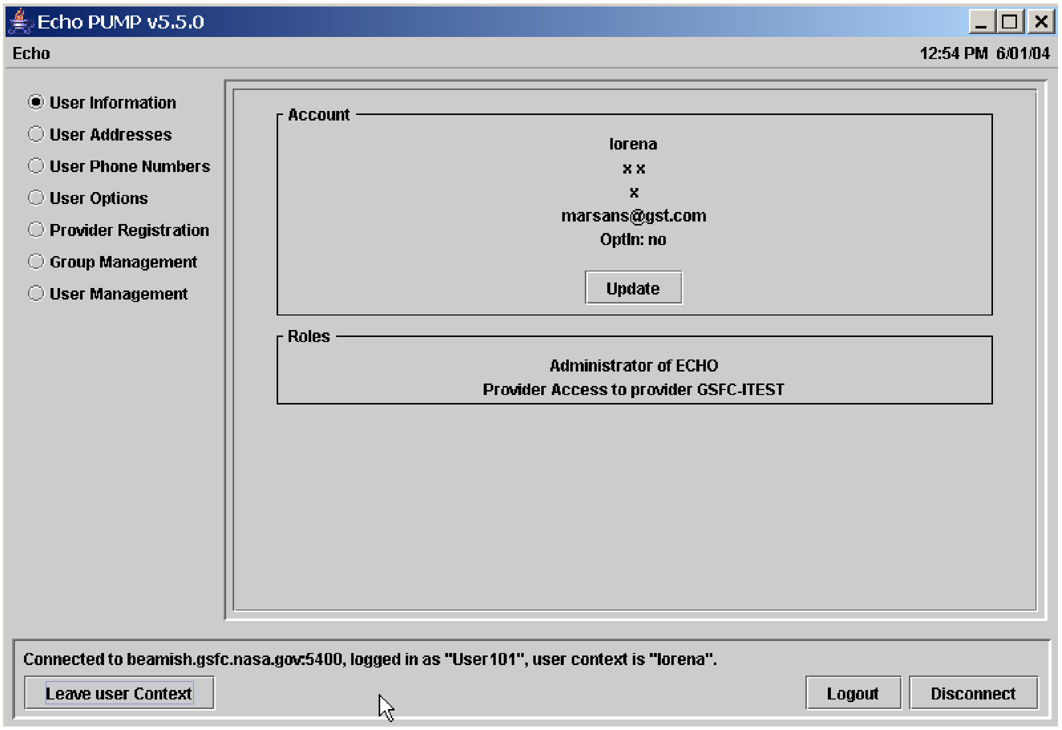

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101".

Figure 6-8. Set user context.



lorena
x x
x
marsans@gst.com
OptIn: no

Administrator of ECHO
Provider Access to provider GSFC-ITEST

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", user context is "lorena".

Figure 6-9. Leave user context.

Note: Working as a different user, you will only have the capabilities allowed to that user.

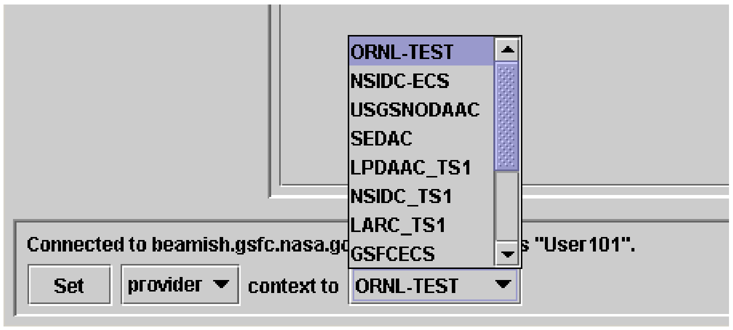
6.2.6 Setting Provider Context

An ECHO Provider participates with ECHO to provide Earth Science Data in the form of its metadata holdings. This metadata is available for search and retrieval as well as navigation and discovery. Data Providers also provide a mechanism to allow Science Users to access the data holdings. ECHO allows Providers the capability to manage access to their data.

To perform any data management, you can only do it on behalf of a provider, that is login as a user using a provider context. You can only do this if you are granted permission to work on behalf of that provider.

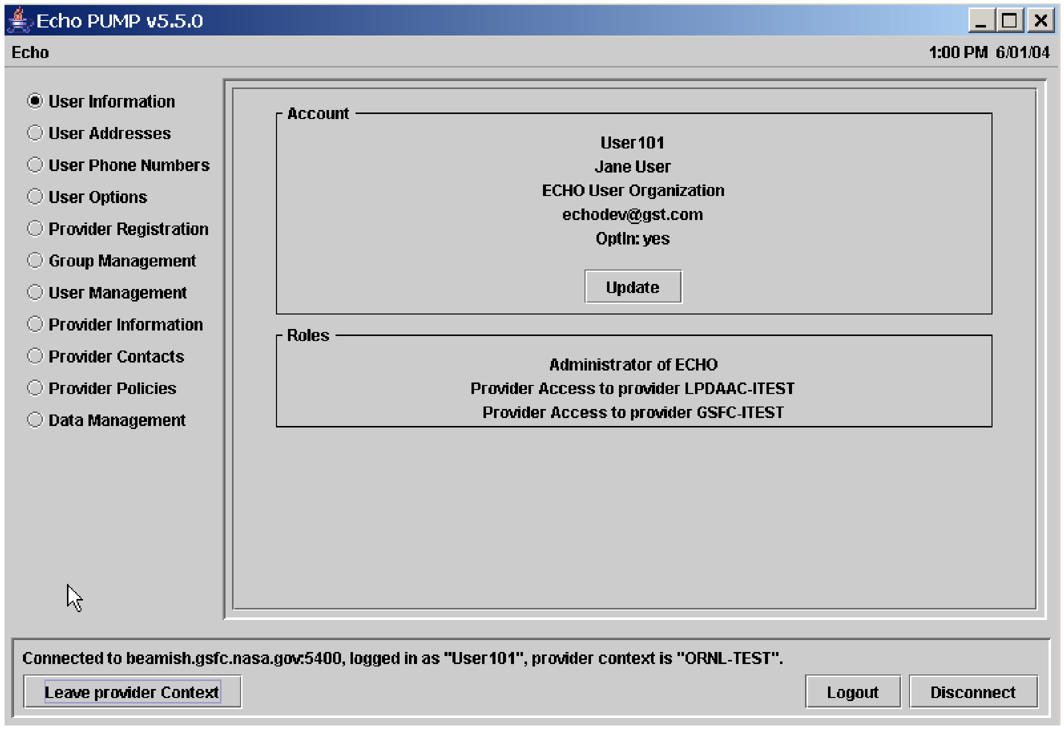
To set the provider context for a provider for whom you have been granted access privileges:

1. Go to the server session area located at the bottom of the screen and select the "provider" option from the first pull-down button. Select the provider you want from the second pull-down button (Figure 6-10).
2. Click the "Set" button on the left. The status bar will be updated and the navigator will have options for data management that only providers can perform (Figure 6-11).
3. Working in the role of a provider, you will have the capabilities allowed to providers. To leave the provider context, click the "Leave Provider Context" button.



Connected to beamish.gsfc.nasa.go ... s "User101".

Figure 6-10. Set provider context.



Account

User101
Jane User
ECHO User Organization
echodev@gst.com
OptIn: yes

Roles

Administrator of ECHO
Provider Access to provider LPDAAC-ITEST
Provider Access to provider GSFC-ITEST

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

Figure 6-11. Leave provider context.

6.3 DATA AND USER MANAGEMENT

6.3.1 Data Management

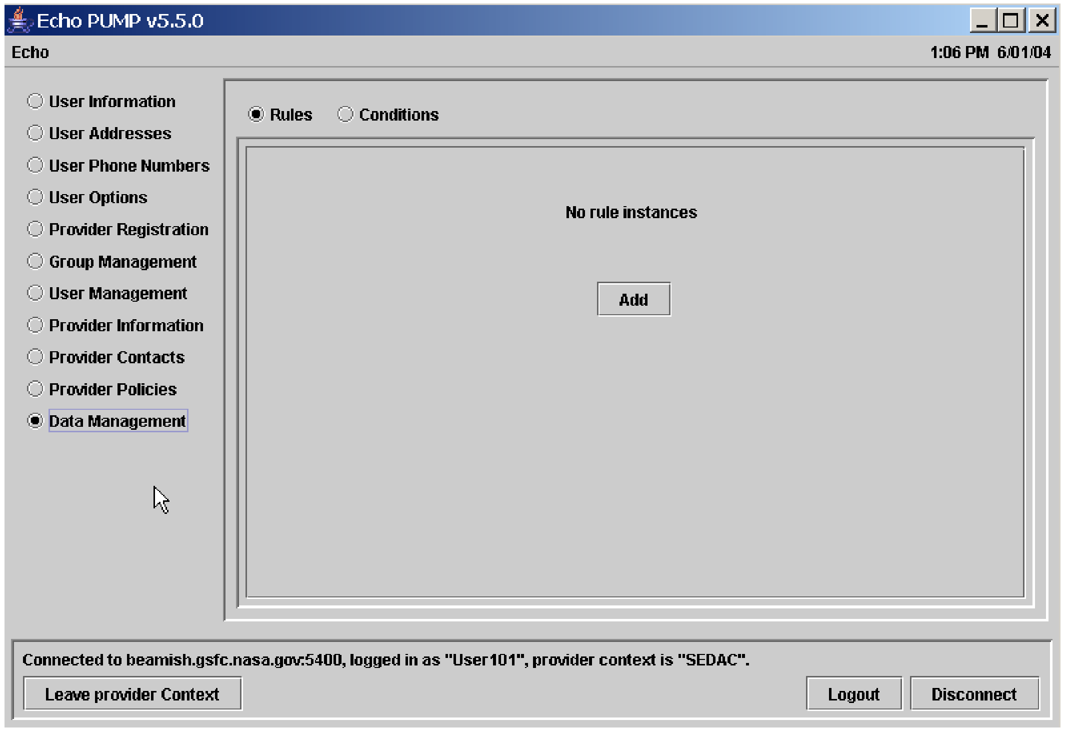
To control access to metadata, providers use the Data Management Service. This service restricts or permits users of the ECHO system to take action upon metadata. Actions include viewing, browsing, and ordering data defined using metadata. Providers manage data by the use of "conditions," which specify metadata characteristics. Example conditions are: the month of January, 30 days old, qaFlag2 value 3.

When a Condition, Comparator, Action, and Data Holding come together, a rule is formed. The rule can be either restriction or permission. For example, the rule "restrict viewing of all MODIS metadata less than 30 days old" is composed of the following:

1. Condition: 30 days old;
2. Comparator: less than;
3. Action: view;
4. Data Holding: instrument MODIS.

If a rule is a restriction, it applies to the entire ECHO user community. If a rule is a permission, it applies only to members of the group specified.

To use the Data Management Service with PUMP, after you login as a user, you have to set the context to the provider you will be working for. Once you have the provider context set, you will be presented with options for providers, select the "Data Management" option from the navigator menu (Figure 6-12).

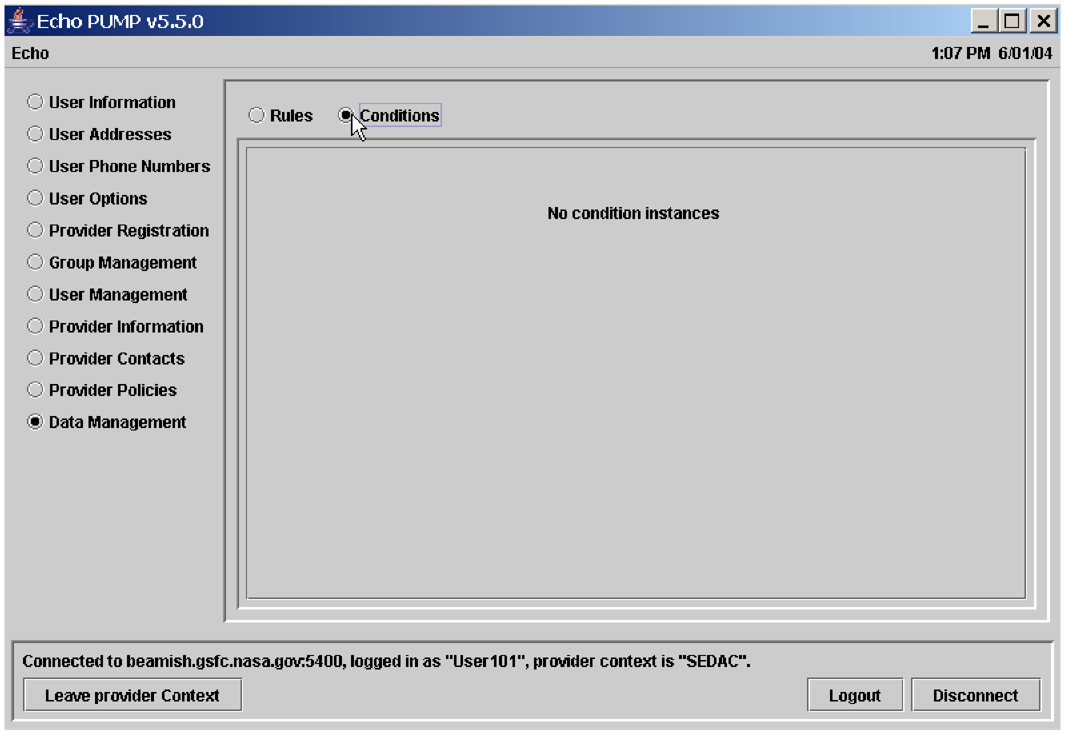


No rule instances

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "SEDAC".

Figure 6-12. No rules.

Notice that there are two option buttons on top, "Rules" and "Conditions". If you toggle between them, they display a list of rules or conditions (Figure 6-13 above and Figure 6-14 below)


Echo PUMP v5.5.0

Echo 1:07 PM 6/01/04

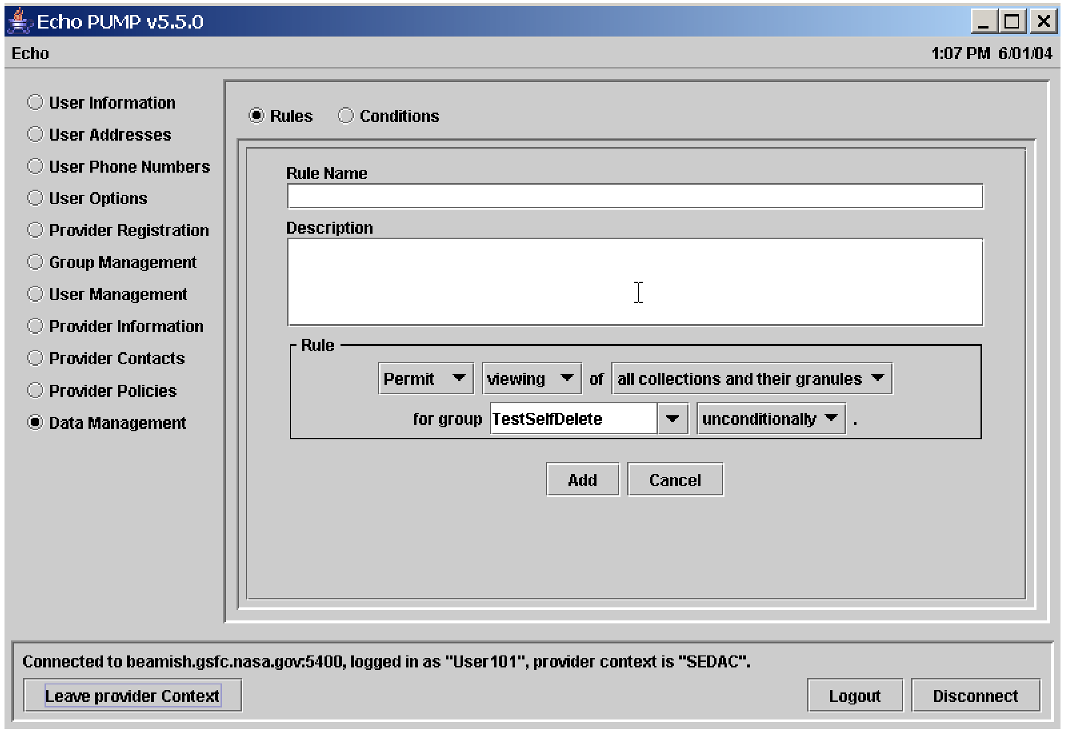
- User Information
- User Addresses
- User Phone Numbers
- User Options
- Provider Registration
- Group Management
- User Management
- Provider Information
- Provider Contacts
- Provider Policies
- [✓ Data Management]

Rules [✓ Conditions]

No condition instances

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "SEDAC".

Figure 6-13. No conditions.


Echo PUMP v5.5.0

Echo 1:07 PM 6/01/04

- User Information
- User Addresses
- User Phone Numbers
- User Options
- Provider Registration
- Group Management
- User Management
- Provider Information
- Provider Contacts
- Provider Policies
- [✓ Data Management]

[✓ Rules] Conditions

Rule Name

Description

Rule

Permit viewing of all collections and their granules

for group TestSelfDelete unconditionally .

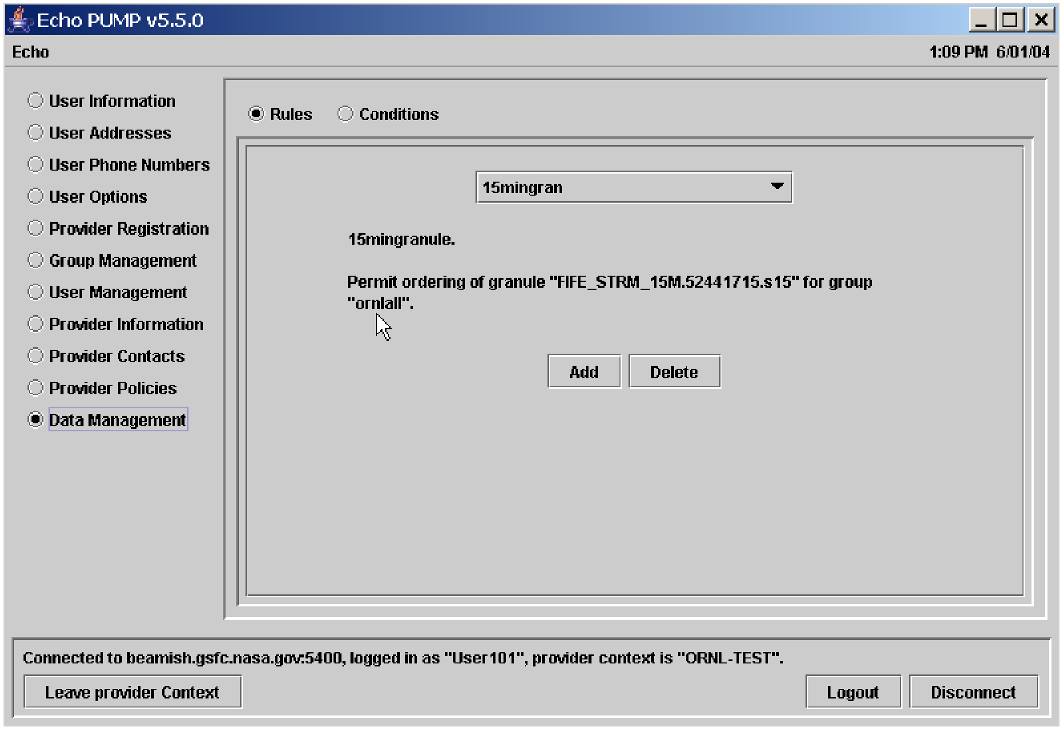
Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "SEDAC".

Figure 6-14. Add rule.

To add a rule:

1. Select the Rules option button, then click on the “Add” button to get the add rule screen (Figure 6-14).
2. Fill in the input fields for “Rule Name” and “Description”.
3. Set the rule by selecting the proper pull down buttons. Select the type of rule from the dropdown menu, options are "permit" or "restrict".
4. Select the action from the dropdown menu, options are "viewing" or "ordering".
5. Select the data holding, options are "all collections and their granules", "all collection granules", "collection", "all granules", "granule".
6. Select the comparator and the condition. Depending on the data holding selected, you will have different selections for comparator and condition.
7. Click the "Add" button. You will get the Display Rules Screen with the rule you just created (Figure 6-15).

Once you have added the rule, the system will automatically create a Condition that can be seen in the Conditions listing (Figure 6-16).

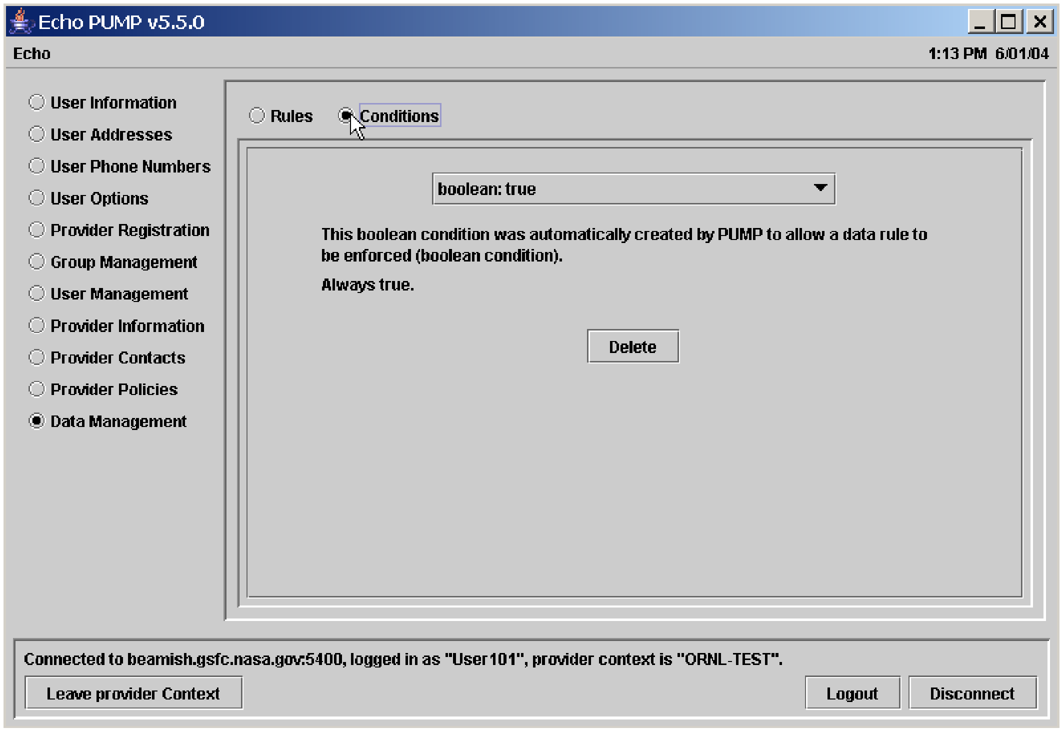


15mingranule.

Permit ordering of granule "FIFE_STRM_15M.52441715.s15" for group "ornlall".

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

Figure 6-15. Display new rule.



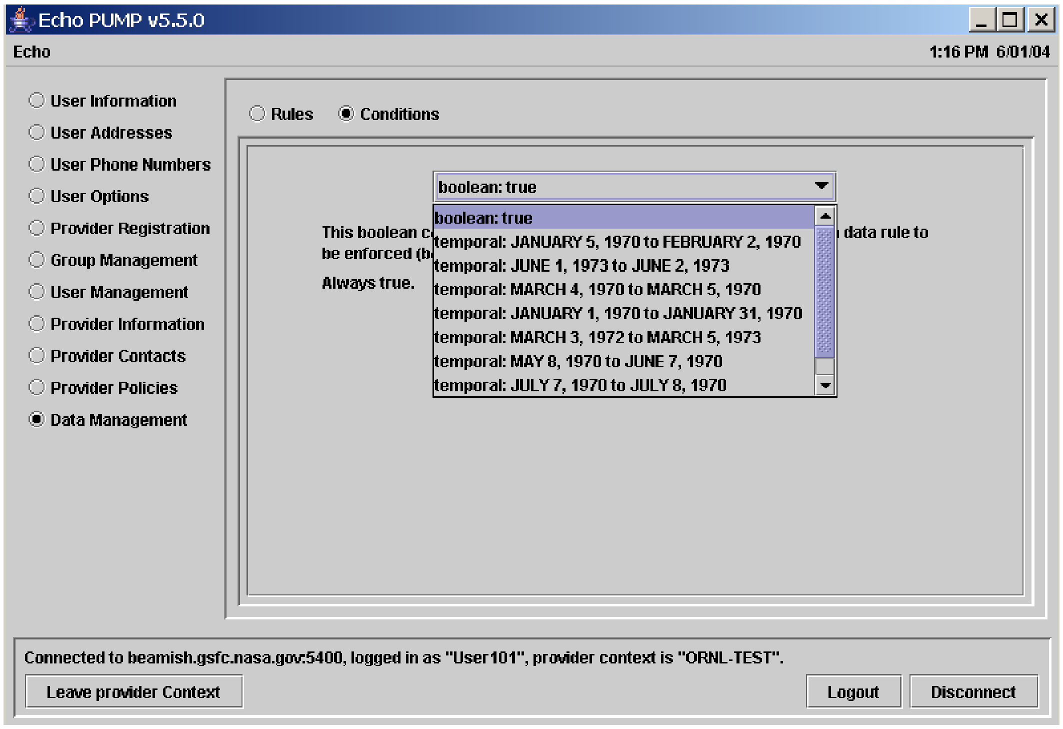
This boolean condition was automatically created by PUMP to allow a data rule to be enforced (boolean condition).

Always true.

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

Figure 6-16. New condition generated by PUMP.

If you have several conditions, to view a specific one, select it from the dropdown menu (Figure 6-17).



boolean: true
temporal: JANUARY 5, 1970 to FEBRUARY 2, 1970
temporal: JUNE 1, 1973 to JUNE 2, 1973
temporal: MARCH 4, 1970 to MARCH 5, 1970
temporal: JANUARY 1, 1970 to JANUARY 31, 1970
temporal: MARCH 3, 1972 to MARCH 5, 1973
temporal: MAY 8, 1970 to JUNE 7, 1970
temporal: JULY 7, 1970 to JULY 8, 1970

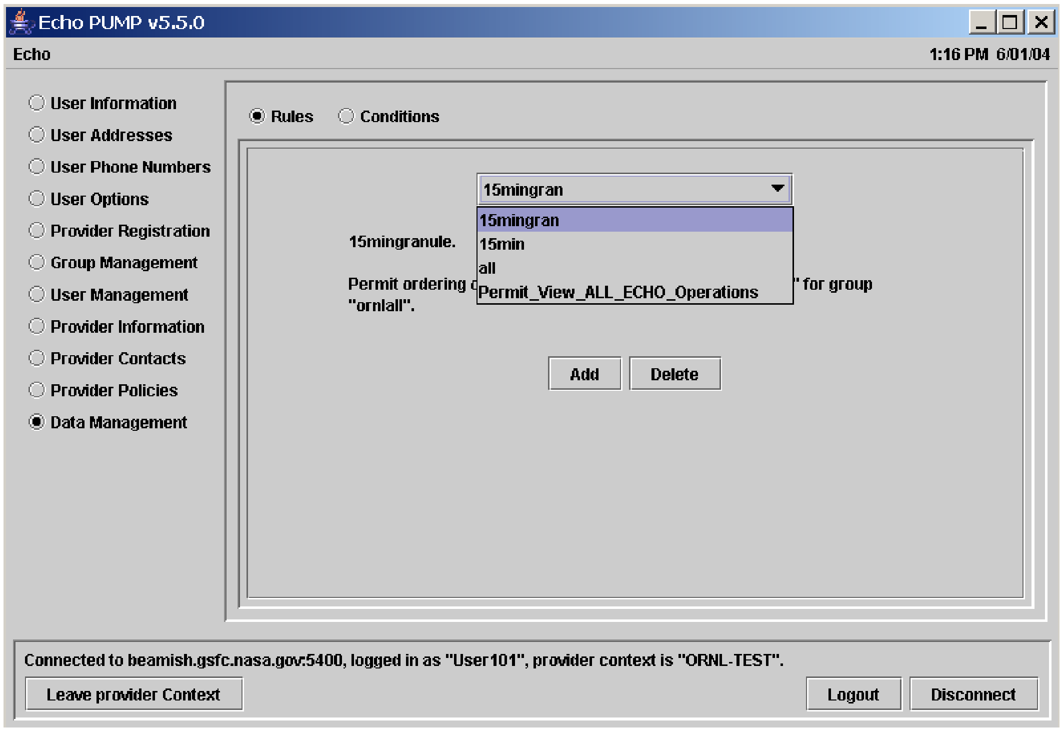
This boolean c… data rule to be enforced (b…

Always true.

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

Figure 6-17. Many conditions.

Once you've added several rules, to view a specific one select the Rules option button, then select the needed rule from the dropdown menu (Figure 6-18). The chosen rule will be displayed.



15mingranule.

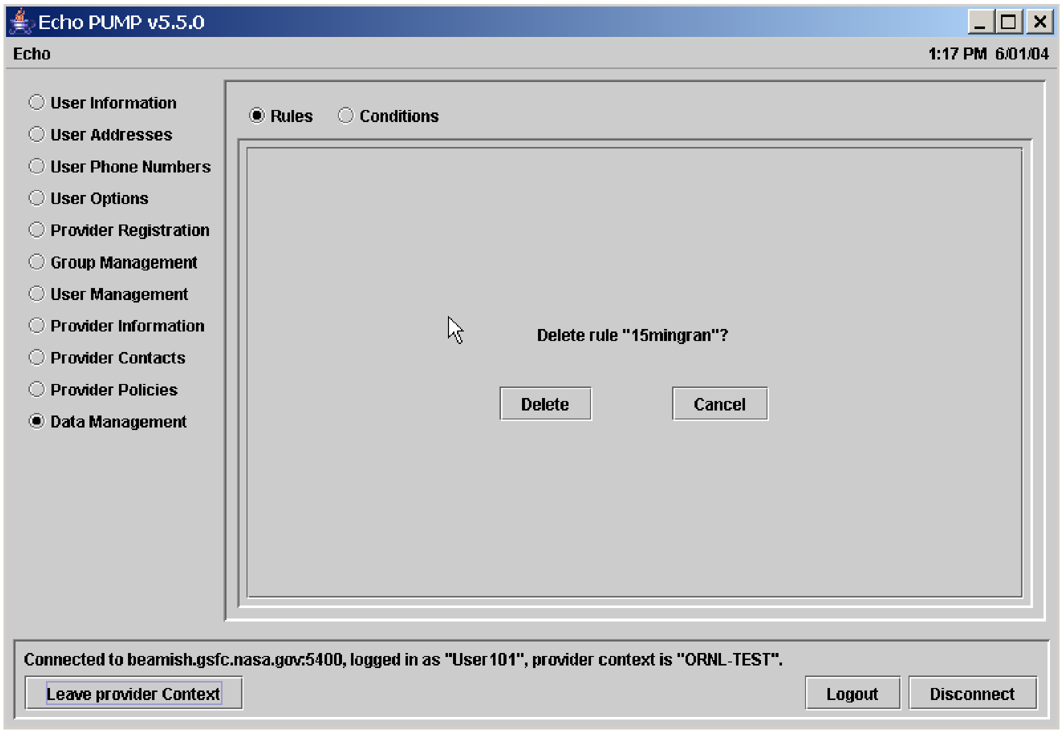
Permit ordering … " for group "ornlall".

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

Figure 6-18. Many rules.

To delete a rule:

1. Display the rule you want to delete, click on the "Delete" button. You will get a screen prompting you to confirm the deletion (Figure 6-19).
2. Click on the "Delete" button. You will get a message indicating successful deletion of the rule.



Delete rule "15mingran"?

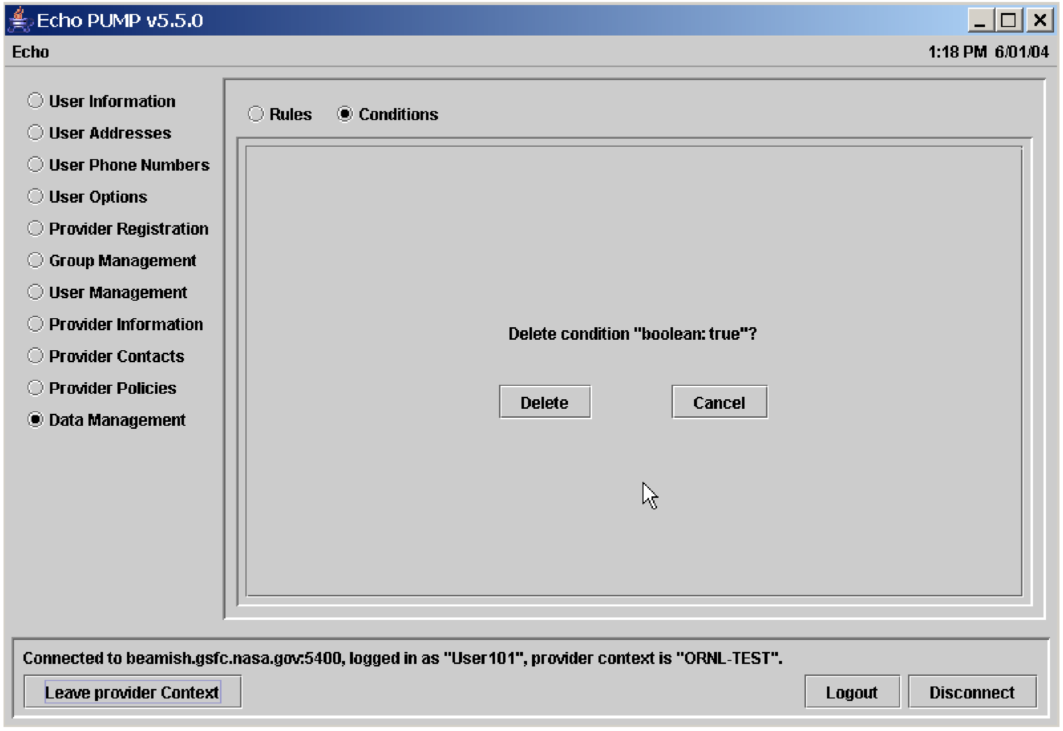
Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

Figure 6-19. Delete rule.

To delete a Condition:

1. Select the condition you want to delete
2. Click on the "Delete" button. You will get a screen prompting you to confirm the deletion (Figure 6-20).
3. Click on the "Delete" button. You will get a message indicating successful deletion of the condition.

Note: You can cancel a transaction at any time by clicking on the "Cancel" button from any of the working screens.



Delete condition "boolean: true"?

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

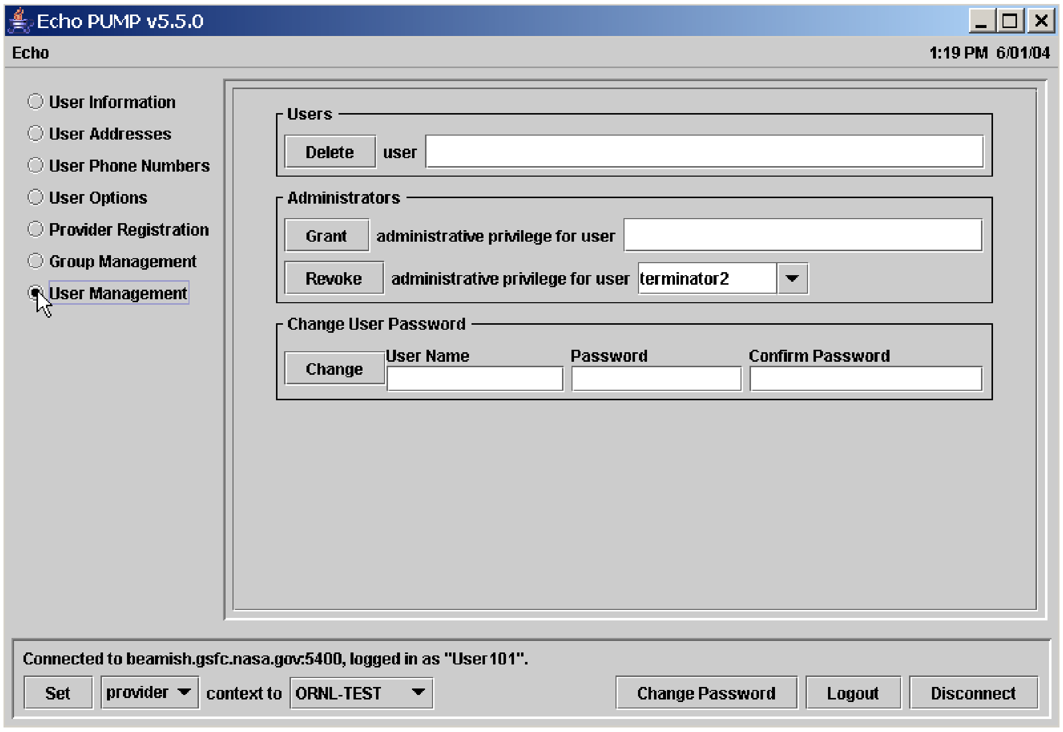
Figure 6-20. Delete condition.

6.3.2 User Management

As an administrative user, you can delete users and grant and/or revoke administrative privileges to users. To perform any of these activities:

1. Select the User Management button on the navigator to get the User Management screen (Figure 6-21).
2. Enter the user name for the user you want to manage in the appropriate field, depending on the activity you need to perform.
3. Click the button for that activity – "Delete", "Grant", or "Revoke". You will be prompted for the action to make sure is the correct action (Figure 6-22).
4. At the prompt, click the "Grant" button to finish the transaction, or the "Cancel" button to cancel. When the transaction has completed, you will get a screen with the message indicating completion, then you will return to the User Management screen.

Note: You can exit a transaction at any time by clicking the "Cancel" button from any of the working screens.



Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101".

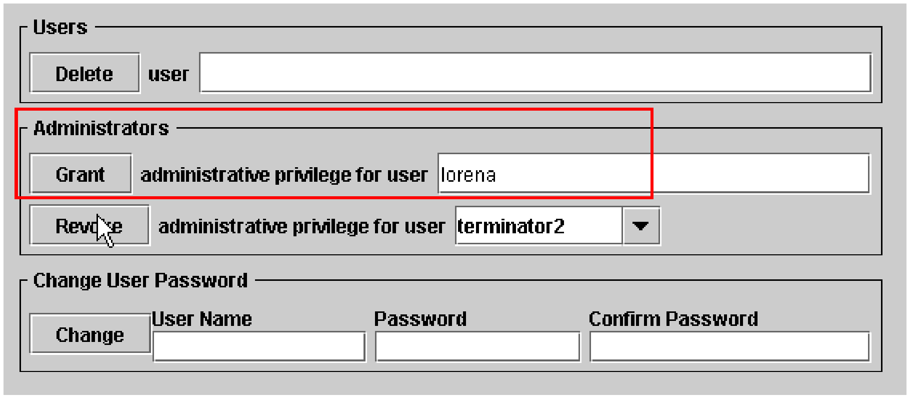
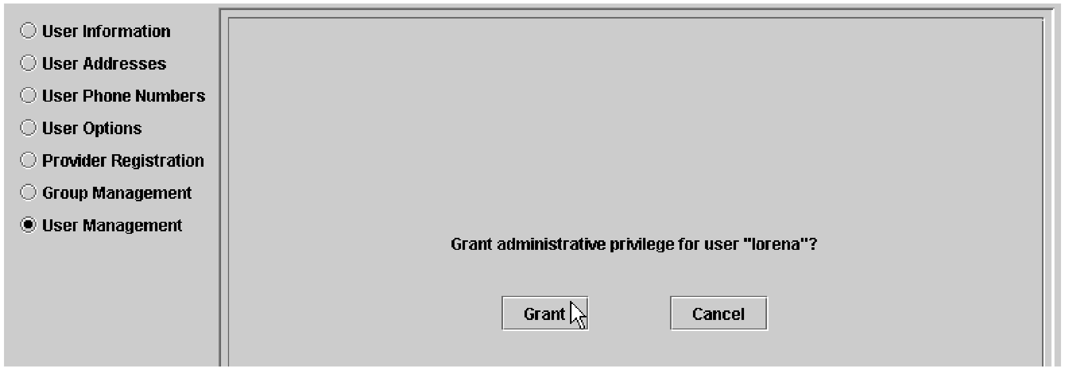


Figure 6-21. User management.



Grant administrative privilege for user "lorena"?

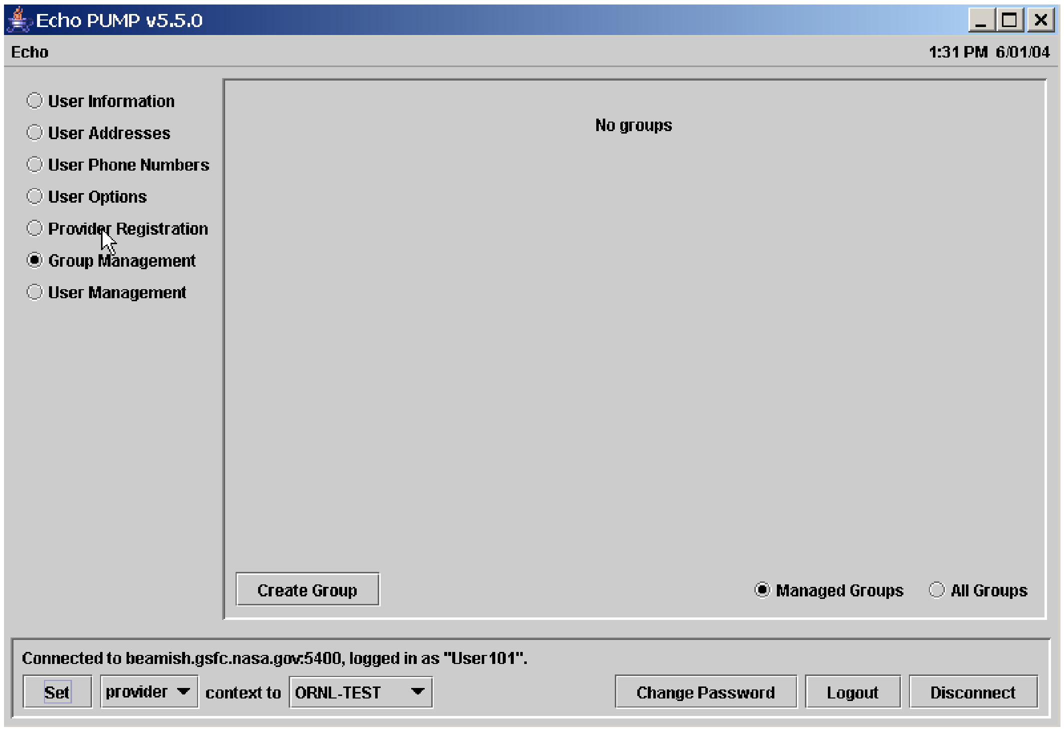
Figure 6-22. Grant administrative privilege.

6.3.3 Group Management

As a user acting on behalf of a provider (or as an administrative user), you can manage groups. The Group Management Service provides all activities that an ECHO user needs to maintain and manage groups of users. Groups are aggregations of ECHO registered users, and there are rules governing the membership actions that can be invoked on a group. Each group consists of one owner, and one or more managers. A group manager is responsible for adding and removing ECHO registered users to the group. A group manager is also responsible for the management of the group managers. Groups are used to communicate among groups of users and their managers. Groups are also used in the Data Management Service when permissions are created on metadata.

To create a group:

1. Select the Group Management service button from the navigator to get the Group Management screen (Figure 6-23 and Figure 6-24).
2. Click the Create Group button.



No groups

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101".

Figure 6-23. No groups.

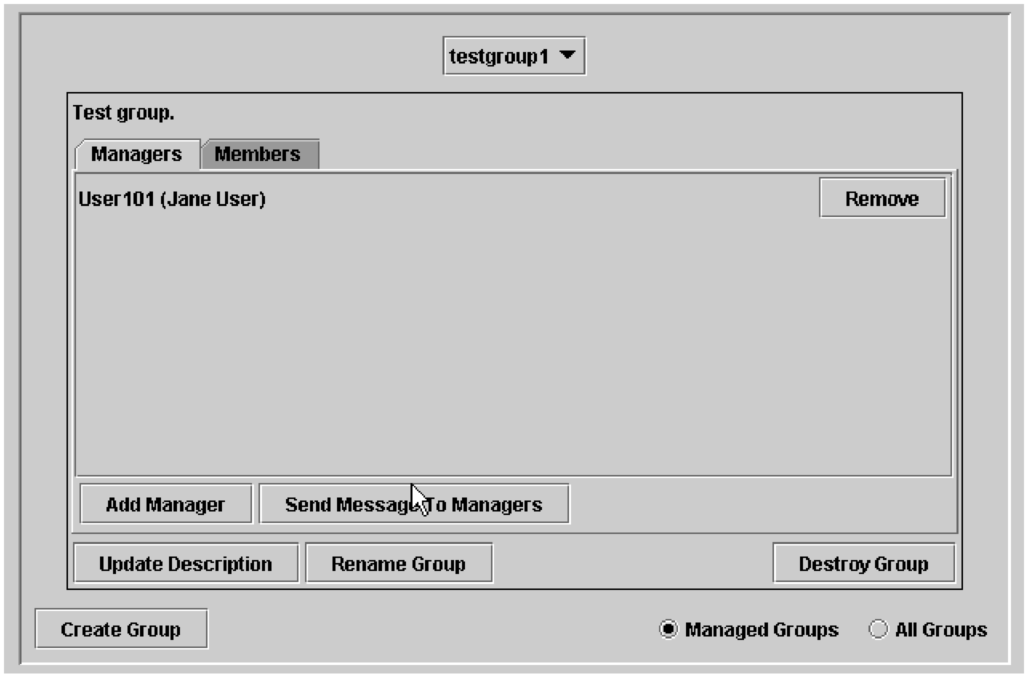


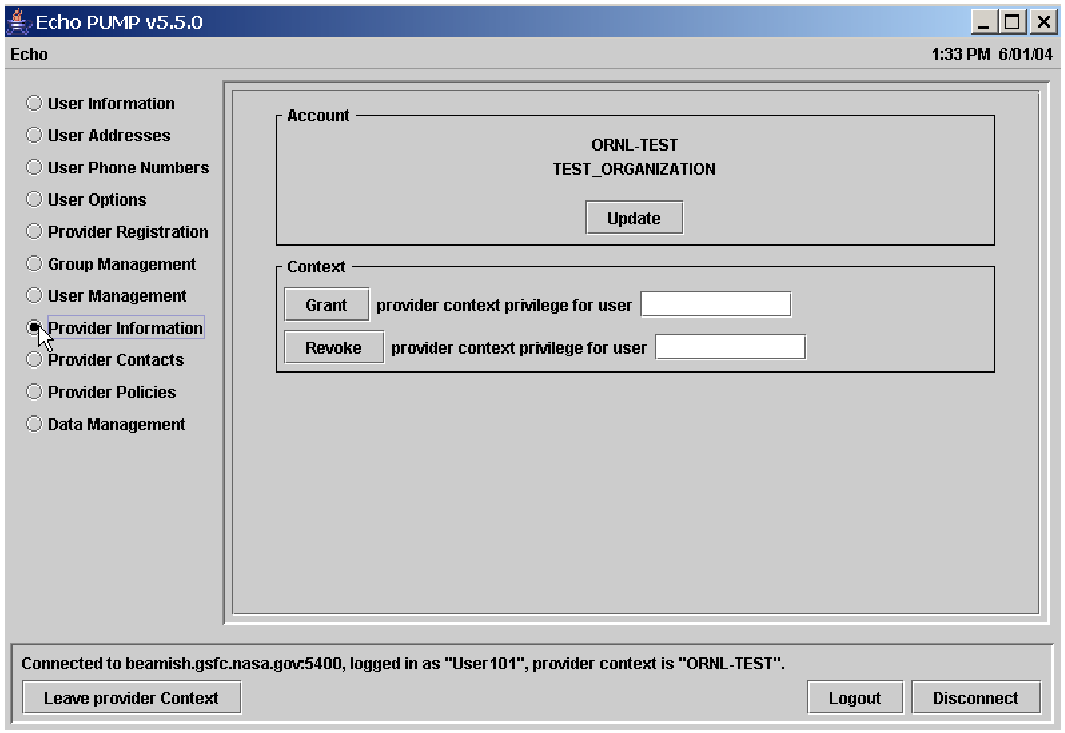
Figure 6-24. Create Group.

6.4 UPDATING INFORMATION IN ECHO

6.4.1 Provider Information

To view the provider information:

1. Select the Provider Information button from the navigator on the upper left part of the screen; you will get the following screen (Fig 6-25).



ORNL-TEST
TEST_ORGANIZATION

Grant provider context privilege for user

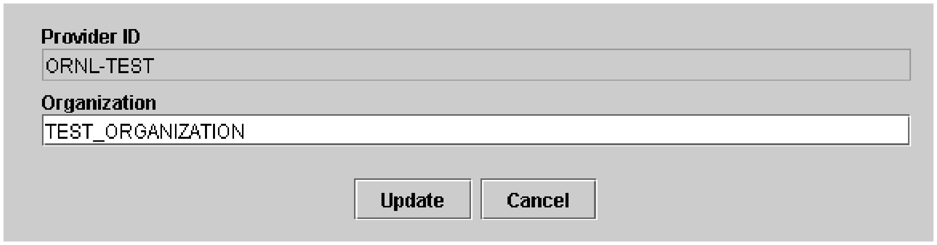
Revoke provider context privilege for user

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

Figure 6-25. Initial Provider Information Screen

To change the information:

1. Click on the "Update" Button; you will get the Information edit screen. (Figure 6-26).
2. Fill in the Organization field. Click on "Update" for this change to take effect. You will get a message indicating a successful update. The page will reload to the uneditable state with the new values displayed.



Provider ID: ORNL-TEST

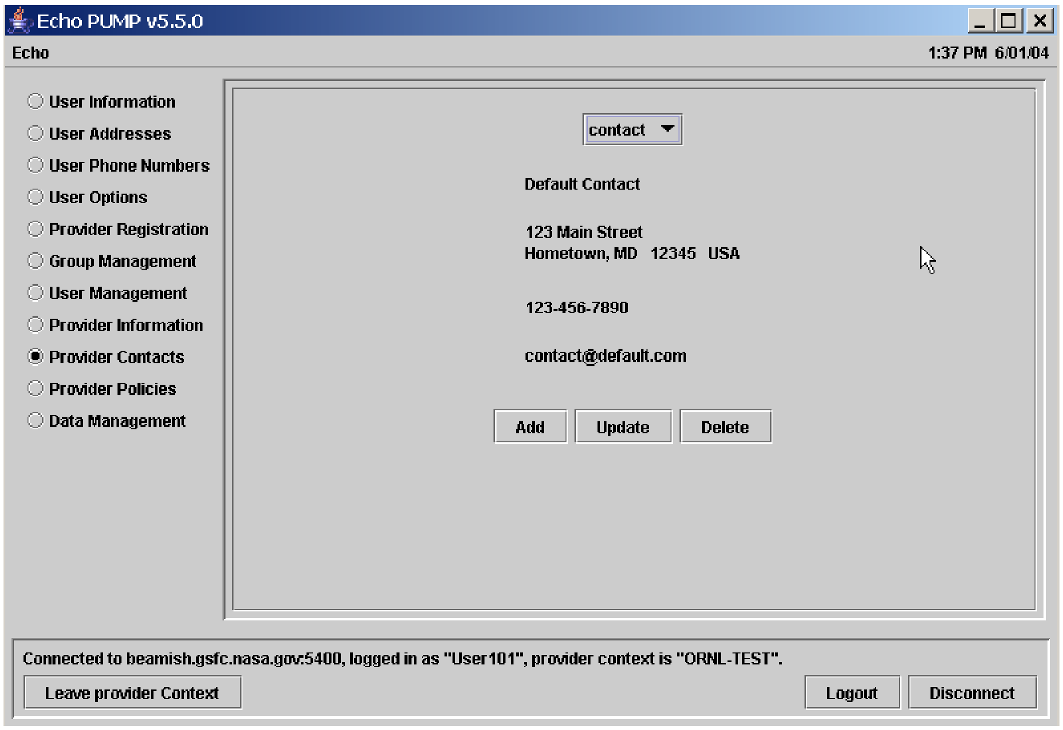
Organization: TEST_ORGANIZATION

Figure 6-26. Editable Provider Information Screen

6.4.2 Provider Contacts

To view the provider contact information:

1. Select the Provider Contacts button from the navigator on the upper left part of the screen; you will get the following screen (Figure 6-27).



Default Contact

123 Main Street
Hometown, MD 12345 USA

123-456-7890

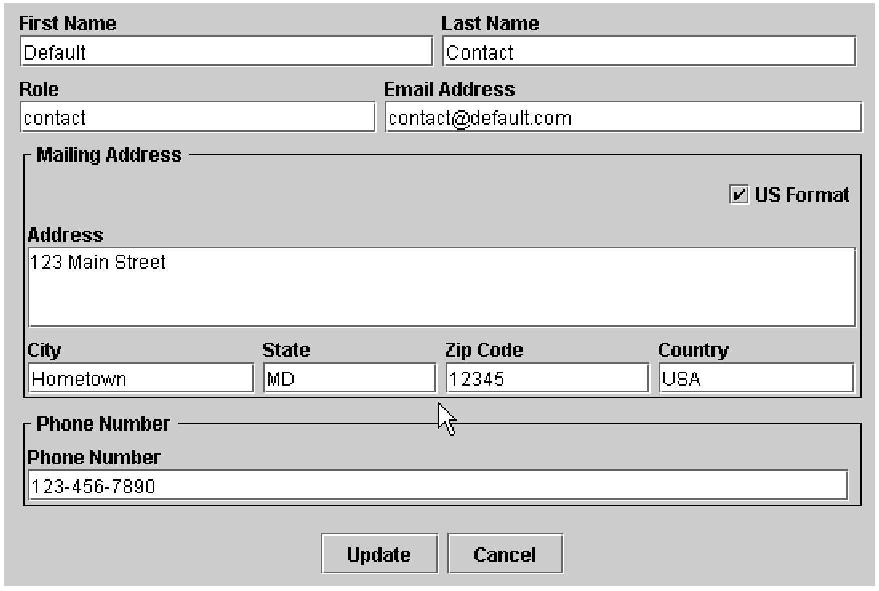
contact@default.com

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

Figure 6-27. Initial Provider Contacts Screen

To change the contact information:

1. Click on the "Update" Button; you will get the Contact edit screen. (Figure 6-28).
2. Fill in the Name/Mail/Phone fields. Click on "Update" for this change to take effect. You will get a message indicating a successful update. The page will reload to the uneditable state with the new values displayed.



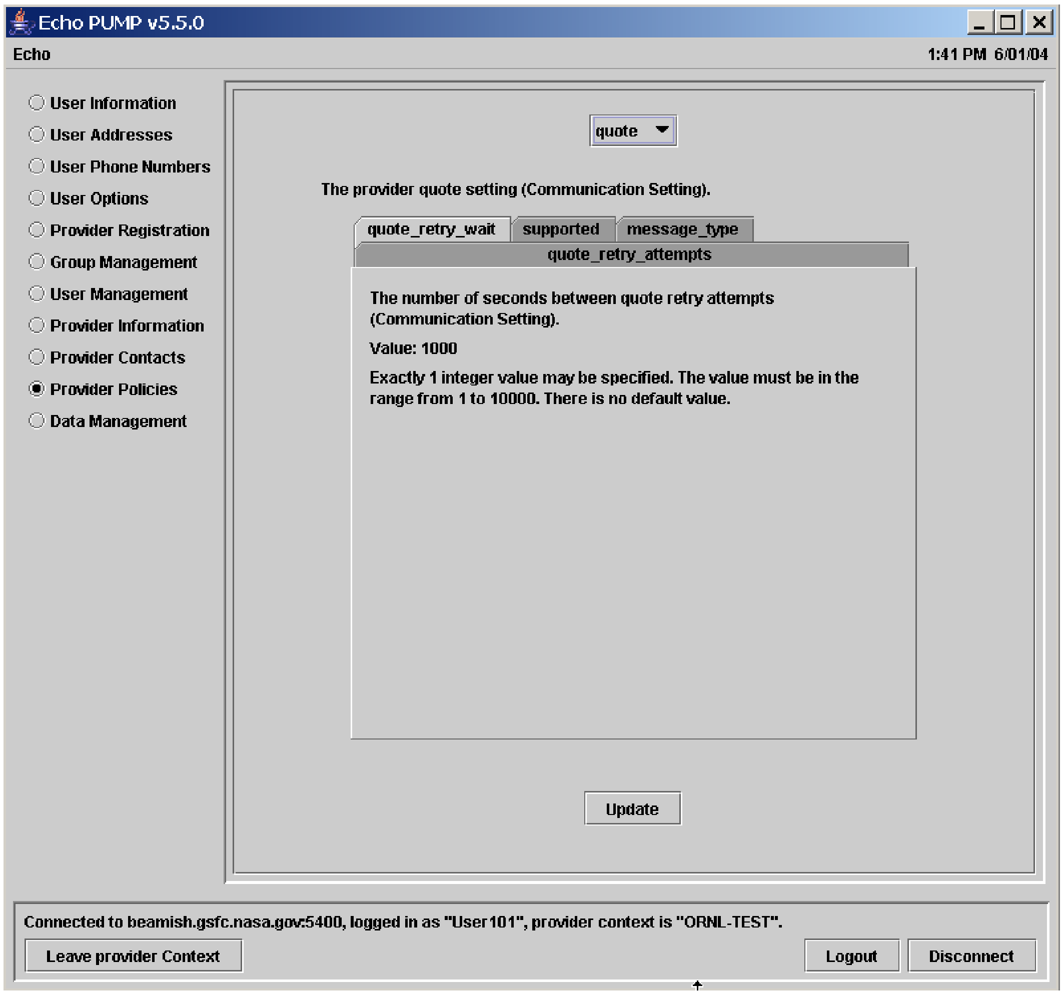
First Name: Default
Last Name: Contact
Role: contact
Email Address: contact@default.com
[✓] US Format
Address: 123 Main Street
City: Hometown
State: MD
Zip Code: 12345
Country: USA
Phone Number: 123-456-7890

Figure 6-28. Editable Provider Contact Screen

6.4.3 Provider Policies

To view the policies:

1. Select the Provider Policies button from the navigator on the upper left part of the screen; you will get the following screen (Figure 6-29).



The provider quote setting (Communication Setting).

The number of seconds between quote retry attempts (Communication Setting).

Value: 1000

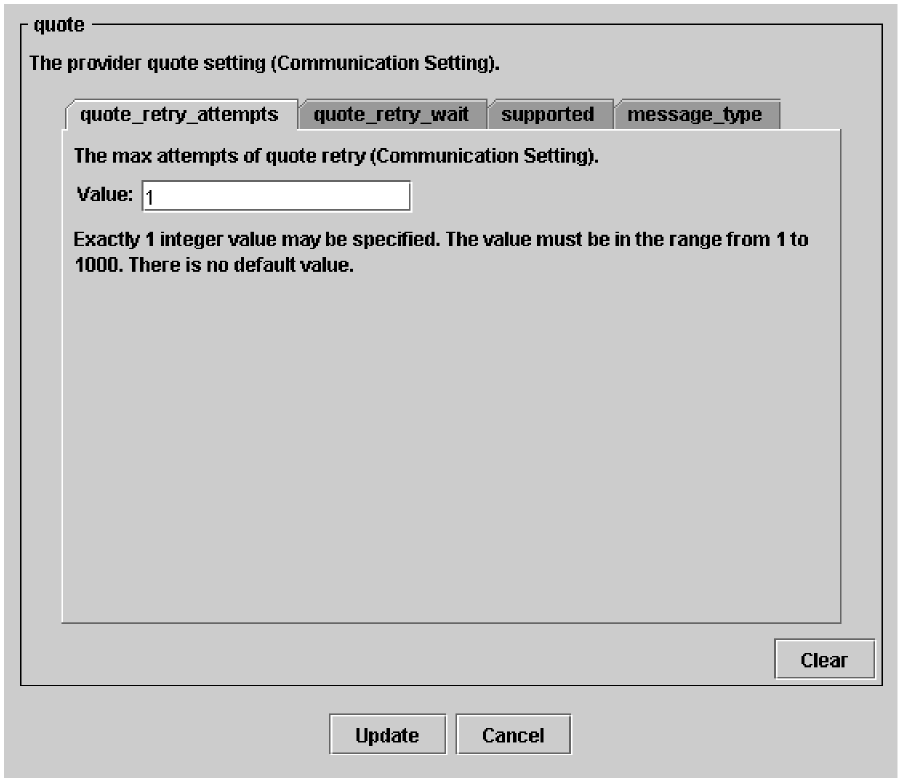
Exactly 1 integer value may be specified. The value must be in the range from 1 to 10000. There is no default value.

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101", provider context is "ORNL-TEST".

Figure 6-29. Display Provider Policies screen

To change the policies:

1. Click on the “Update” Button; you will get the Policy edit screen. (Figure 6-30).
2. Fill in the fields appropriate for your site. Click on “Update” for these changes to take effect. You will get a message indicating a successful update. The page will reload to the uneditable state with the new values displayed.


quote

The provider quote setting (Communication Setting).

The max attempts of quote retry (Communication Setting).

Value: 1

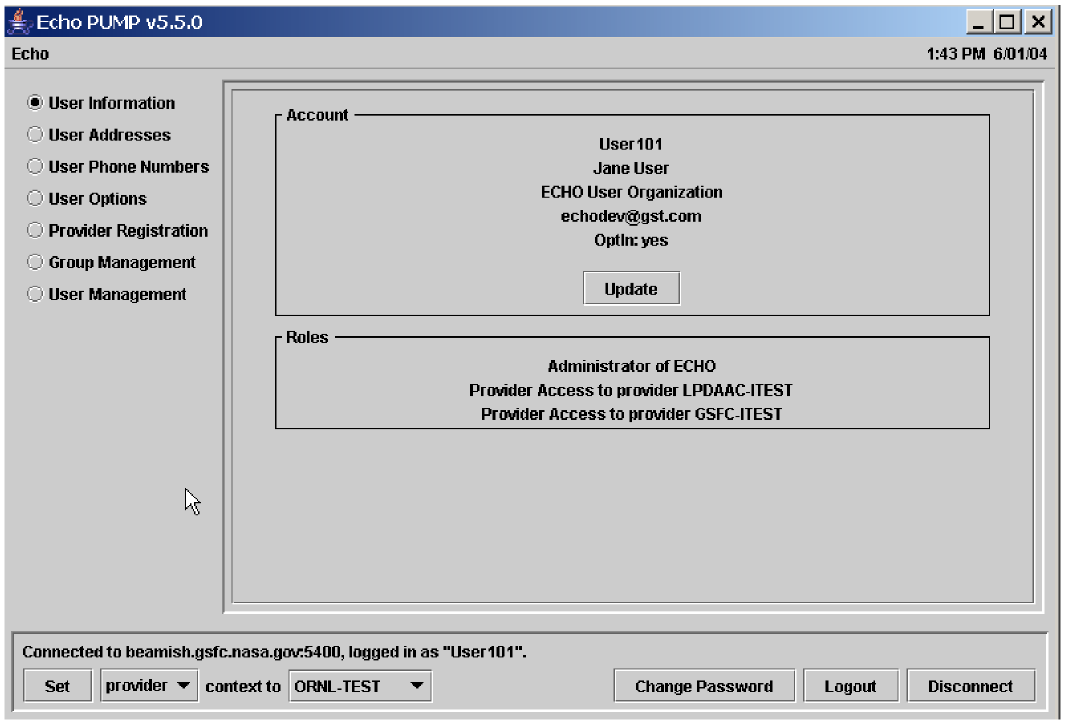
Exactly 1 integer value may be specified. The value must be in the range from 1 to 1000. There is no default value.

Figure 6-30. Editable Provider Policies Screen

6.4.4 User Information

To view your user information:

1. Select the User Information button from the navigator on the upper left part of the screen; you will get the user information screen (Figure 6-31).



User101
Jane User
ECHO User Organization
echodev@gst.com
OptIn: yes

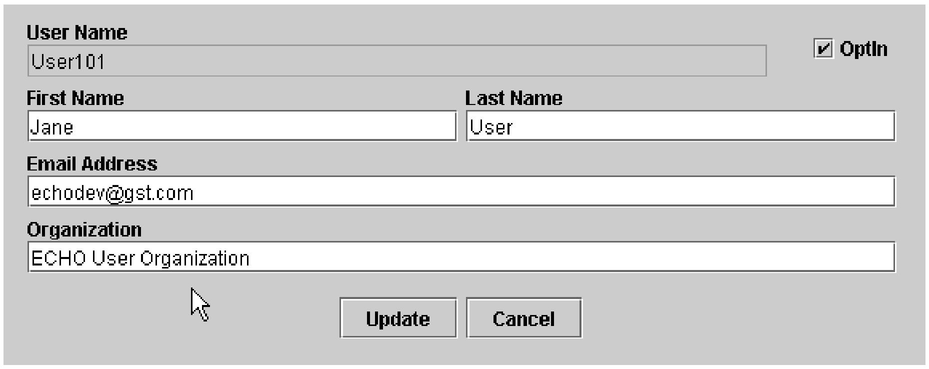
Administrator of ECHO
Provider Access to provider LPDAAC-ITEST
Provider Access to provider GSFC-ITEST

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101".

Figure 6-31. Display user information.

To change the information:

1. Click on the "Update" button; you will get the user information edit screen (Figure 6-32).
2. Fill in the fields and click the "Update" button. You will get a message indicating successful update of your user information. The user information screen will display the updated values.



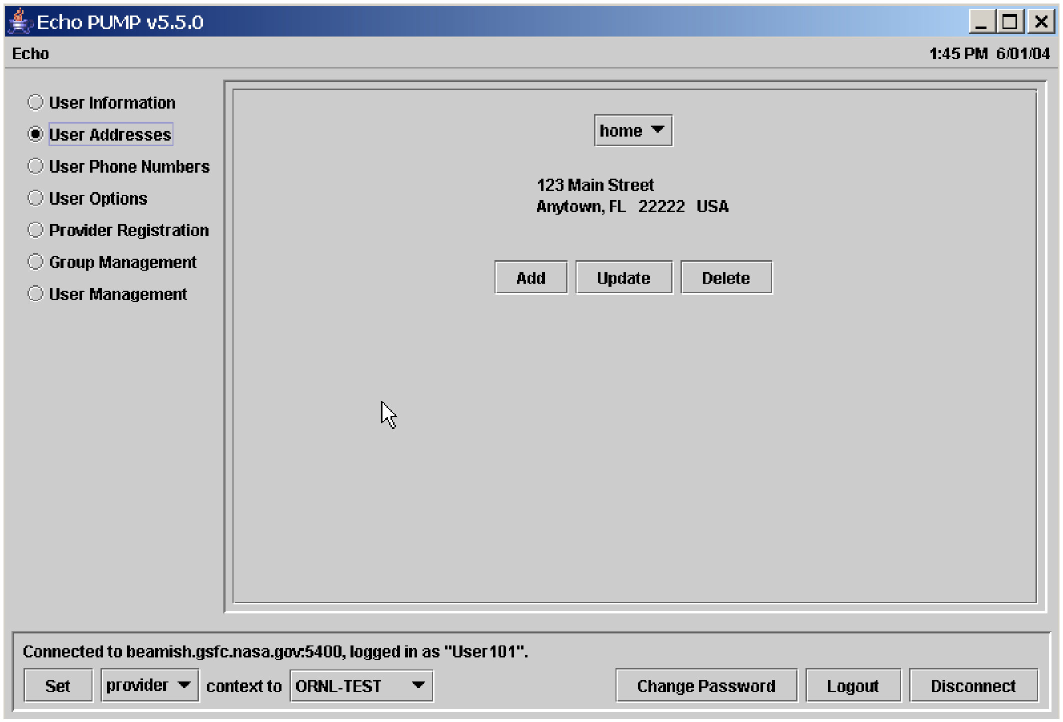
User Name	User101
OptIn	✓
First Name	Jane
Last Name	User
Email Address	echodev@gst.com
Organization	ECHO User Organization

Figure 6-32. Edit user information.

Note: You can exit a transaction at any time by clicking the "Cancel" button from any of the working screens.

6.4.5 User Addresses

To view your user address information, select the User Addresses button from the navigator to get the user addresses display screen (Figure 6-33).



123 Main Street
Anytown, FL 22222 USA

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101".

Figure 6-33. Display user address.

To add a new user address:

1. Click the Add button to get add address screen (Figure 6-34).
2. Fill in the fields and click on the Add button. You will get a message indicating the successful transaction. The user addresses screen will display the new added information.

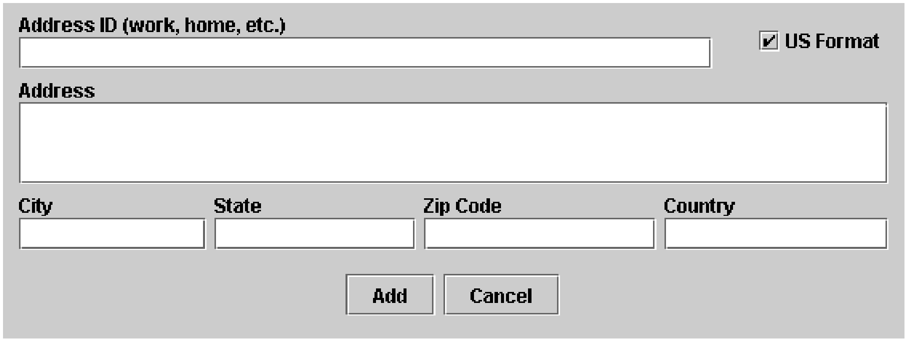
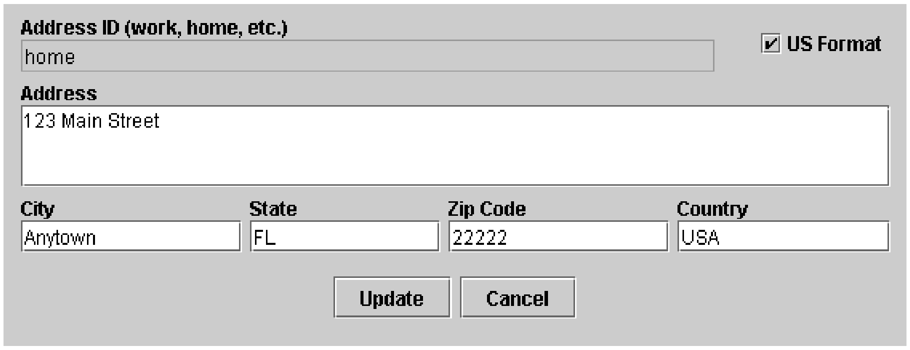


Figure 6-34. Add user address.

To update an existing user address:

1. From the user addresses display screen select the address you want to update.
2. Click the "Update" button to get the user address update screen (Figure 6-35).
3. Modify any fields that need to change, click the "Update" button. You will get a message indicating successful update and the user addresses screen will display the updated values.

Note: You can exit a transaction at any time by clicking the "Cancel" button from any of the working screens.

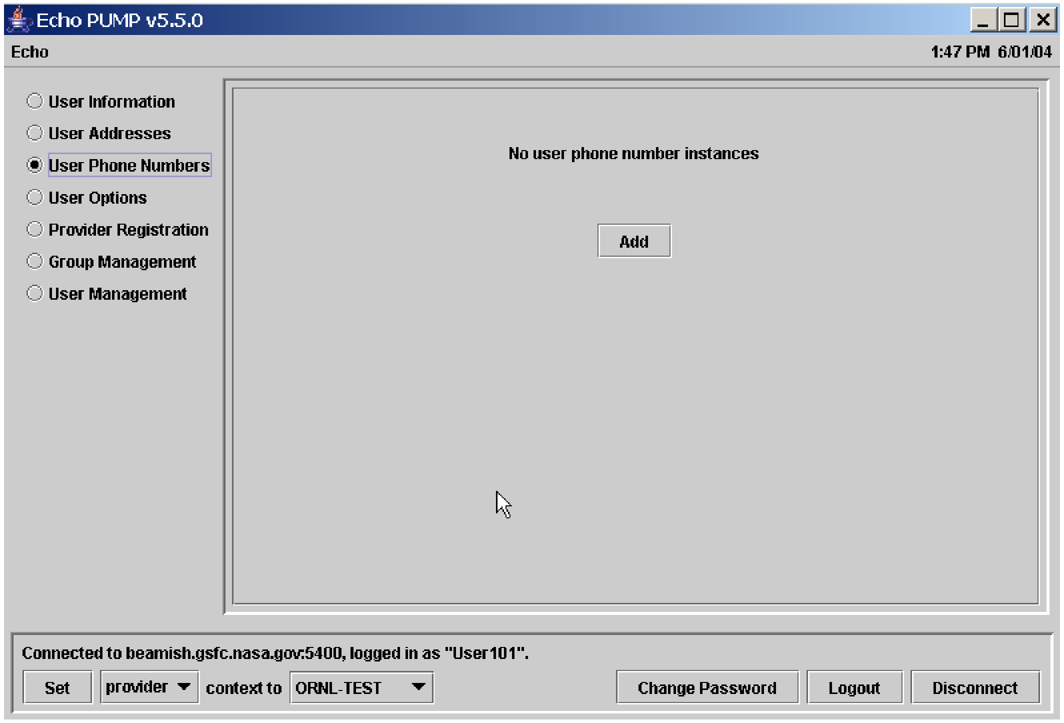

Address ID (work, home, etc.)
home
[✓ US Format]
Address
123 Main Street

City	State	Zip Code	Country
Anytown	FL	22222	USA

Figure 6-35. Update user address.

6.4.6 User Phone Number

To view your user phone numbers information select the User Phone Numbers button from the navigator, you will get the user phone number display screen (Figure 6-36).



● User Phone Numbers

No user phone number instances

Connected to beamish.gsfc.nasa.gov:5400, logged in as "User101".

Set provider context to ORNL-TEST

Figure 6-36. No phone number.

To add a new phone number:

1. Click on the “Add” button to get the user add phone screen (Figure 6-37).
2. Fill in the fields, click on the “Add” button. You will get a message indicating the transaction was successful, then the user phone numbers screen will display the new information (Figure 6-38).

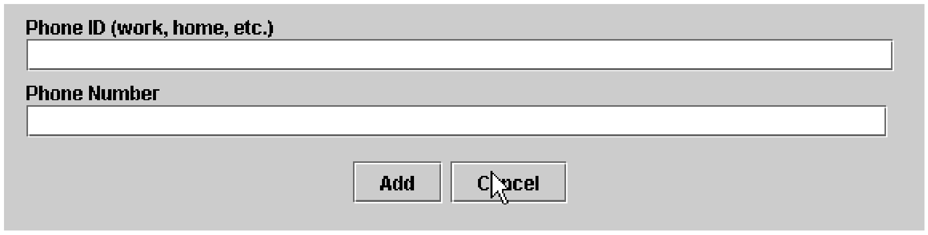


Figure 6-37. Add phone number.

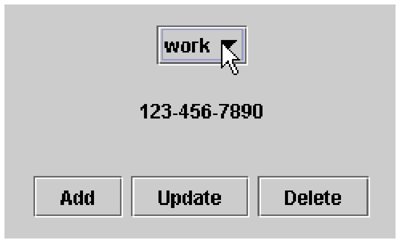
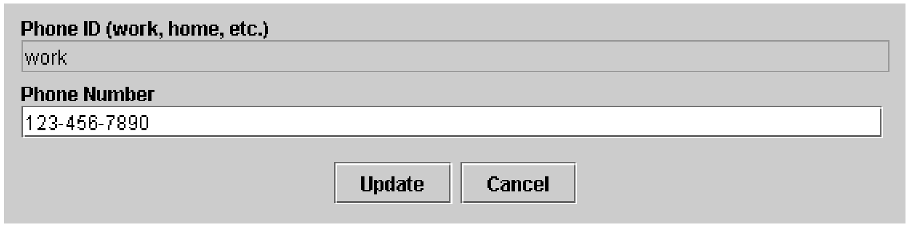


Figure 6-38. Display phone number.

To change the phone number information:

1. From the user phone numbers display screen (Figure 6-39), select the number you want to change using the pull down menu,
2. Click the “Update” button to get the user phone number update screen (Figure 39).
3. Modify the information you want to change, click the “Update button. You will get the user phone numbers display screen with the information updated to reflect your changes.


Phone ID (work, home, etc.): work

Phone Number: 123-456-7890

Figure 6-39. Update user phone number.

To delete a user phone number:

1. From the user phone numbers display screen, select the phone number you want to delete using the pull down menu. Click the “Delete” button. You will get a screen prompting you to confirm the deletion of the number (Figure 6-40).
2. Click on the “Delete” button, you will get the user phone numbers display screen; the deleted phone information will no longer be there.

Note: You can exit a transaction at any time by clicking the “Cancel” button from any of the working screens.

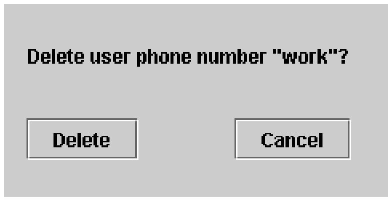

Delete user phone number "work"?

Figure 6-40. Delete user phone number.

6.4.7 Bugs

With v5.5 Pump doesn't create multiple Conditions if the Provider creates multiple rules using the same day.

APPENDIX A: ACRONYMS

API	Application Program Interface
BMGT	Bulk Metadata Generation Tool
DIAL	Data and Information Access Link
DTD	Data Type Definition
ECHO	EOS Clearing House
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
ESDIS	Earth Science Data and Information System
FTP	File Transfer Protocol
GIS	Geographic Information System
GML	Geography Markkup Language
GMT	Greenwich Mean Time
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IMS	Information
ODL	Object Description Language
ORNL	Oak Ridge National Laboratory
PSA	Product Specific Attribute
QA	Quality Assurance
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
UR	Universal Reference
URL	Uniform Resource Locator
XML	eXtensible Markup Language
XSLT	eXtensible Style Language Transformation

APPENDIX B: ANNOTATED OPTIONS EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE CatalogService PUBLIC "-//ECHO CatalogService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/CatalogService.dtd">
 <OrderEntryService>
 <PresentCatalogItemResponse>
 <BooleanResult>
 <BooleanResultType>
  <REQUEST_SUCCEEDED />
  </BooleanResultType>
  </BooleanResult>
 <CatalogItem>
  <CatalogItemID>5054</CatalogItemID>
 <CatalogEntryType>
  <PRODUCT />
  </CatalogEntryType>
  <ProviderID>1003</ProviderID>
  <productDescription />
  <ProductListPrice>0.0</ProductListPrice>
<!--
Option Definition For Product 5054
Product 5054 can be ordered using one of three production options (Ancillary, Native, and Production History). Also it can
be received via CD-ROM or FTP Push.
-->
 <Option>
 <ComplexOption>
  <Name>ECS-TEST PKG1</Name>
  <OptionCategory>Package Option</OptionCategory>
<!--
The type of this complex option is 'STRUCTURE'. Because it is a structure, each of the sub-options with a MinOccurs
attribute value of one or more must be filled in.
-->
 <ComplexOptionType>
  <STRUCTURE />
  </ComplexOptionType>
  <Description>Default Packaging Options</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
<!—
This simple option defines three valid primitive values: 'Ancillary Granule', 'Native Granule', and 'Production History
Granule'. One of these values must be selected.
-->
 <SimpleOption>
  <Name>Production Option</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Production Option</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
```

```
   <SimpleValids>
  <ValidPrimitive>Ancillary Granule</ValidPrimitive>
  <ValidPrimitive>Native Granule</ValidPrimitive>
  <ValidPrimitive>Production History Granule
  </ValidPrimitive>
  </SimpleValids>
  </SimpleOption>
 <ComplexOption>
  <Name>Media Type</Name>
  <OptionCategory>Package Option
  </OptionCategory>
<!--
The type of this complex option is 'CHOICE'. Because it is a choice, exactly one of the sub-options defined must be
selected. One of the two media type options, 'CDROM' or 'FTP Push' must be selected.
-->
 <ComplexOptionType>
  <CHOICE />
  </ComplexOptionType>
  <Description>Media Type</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
   <ComplexOption>
  <Name>CDROM</Name>
  <OptionCategory>Package Option</OptionCategory>
 <ComplexOptionType>
  <STRUCTURE />
  </ComplexOptionType>
  <Description>CDROM</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
<!--
This simple option defines three valid String values: 'Gzip', 'Unix',
and 'None'. One of these values must be selected.
-->
 <SimpleOption>
  <Name>Compression</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Compression Type</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
   <SimpleValids>
  <ValidPrimitive>GZip</ValidPrimitive>
  <ValidPrimitive>Unix</ValidPrimitive>
  <ValidPrimitive>None</ValidPrimitive>
  </SimpleValids>
  </SimpleOption>
<!--
This simple option defines two valid String values: 'Rockridge', and
'ISO9660'. One of these values must be selected. This option is also
optional, as the value of MinOccurs is zero.
```

```
-->
 <SimpleOption>
  <Name>DDISTMEDIAFMT</Name>
  <OptionCategory>Package option</OptionCategory>
  <Description>Dist Media Format</Description>
  <MinOccurs>0</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
   <SimpleValids>
   <ValidPrimitive>Rockridge</ValidPrimitive>
  <ValidPrimitive>ISO9660</ValidPrimitive>
  </SimpleValids>
  </SimpleOption>
  </ComplexOption>
 <ComplexOption>
  <Name>FTP Push</Name>
  <OptionCategory>Package Option</OptionCategory>
   <ComplexOptionType>
  <STRUCTURE />
  </ComplexOptionType>
  <Description>FTP Push</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
 <!--
This simple option defines three valid String values: 'Gzip', 'Unix',
and 'None'. One of these values must be selected.
-->
 <SimpleOption>
  <Name>Compression</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Compression Type</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
   <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
   <SimpleValids>
  <ValidPrimitive>GZip</ValidPrimitive>
  <ValidPrimitive>Unix</ValidPrimitive>
  <ValidPrimitive>None</ValidPrimitive>
  </SimpleValids>
  </SimpleOption>
<!--
This simple option defines a default value: 'Rockridge' The default
value will be filled in as part of the parent structure even if this
option is not set explicitly. Alternately, because this option defines
no valid value list, any string value may be supplied.
-->
 <SimpleOption>
```

```
  <Name>DDISTMEDIAFMT</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Dist Media Format</Description>
  <MinOccurs>0</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
   <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
 <Default>
  <Value>Rockridge</Value>
  </Default>
  </SimpleOption>
<!--
Because this option defines no valid value list, any string value may
Be supplied.
-->
 <SimpleOption>
  <Name>FTPHOST</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>FTP Host</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
  </SimpleOption>
<!--
This simple option defines a default value: 'anonymous' The default
value will be filled in as part of the parent structure even if this
option is not set explicitly. Alternately, because this option defines
no valid value list, any string value may be supplied. Finally, the
'Encrypted' field value of 'True' indicates that an encryption
mechanism should be used to secure this String value when sending in a
message.
-->
 <SimpleOption>
  <Name>FTPPASSWORD</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>FTP Password</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>True</Encrypted>
   <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
   <Default>
  <Value>anonymous</Value>
  </Default>
  </SimpleOption>
<!--
Because this option defines no valid value list, any string value may be
```

```
supplied.
-->
 <SimpleOption>
  <Name>FTPPUSHDEST</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>Target Directory</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
   <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
  </SimpleOption>
<!--
This simple option defines a default value: 'anonymous' The default
value will be filled in as part of the parent structure even if this
option is not set explicitly. Alternately, because this option defines
no valid value list, any string value may be supplied.
-->
 <SimpleOption>
  <Name>FTPUSER</Name>
  <OptionCategory>Package Option</OptionCategory>
  <Description>User to log in as</Description>
  <MinOccurs>1</MinOccurs>
  <MaxOccurs>1</MaxOccurs>
  <Encrypted>False</Encrypted>
 <PrimitiveTypeName>
  <String />
  </PrimitiveTypeName>
 <Default>
  <Value>anonymous</Value>
  </Default>
  </SimpleOption>
  </ComplexOption>
  </ComplexOption>
  </ComplexOption>
  </Option>
  </CatalogItem>
  </PresentCatalogItemResponse>
  </OrderEntryService>
```

APPENDIX C: ANNOTATED OPTION SELECTION EXAMPLE

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE OrderEntryService PUBLIC "-//ECHO OrderEntryService (v5.5)//EN"
"http://api.echo.eos.nasa.gov/echo/dtd/OrderEntryService.dtd">
<OrderEntryService>
   <CreateOrderRequest>
      <OrderLineItem>
         <CatalogItemID>
         5054
      </CatalogItemID>
         <quantityOrdered>
         2
      </quantityOrdered>
<!--
Option Selection For Product 5054
Product 5054 can be ordered using one of three production options (Ancillary, Native, and Production History). Also it can
be received via CD-ROM or FTP Push.
-->
         <OptionSelection>
            <ComplexOptionSelection>
<!--
Root option selection node; the Option Selection must fulfill the constraints defined within the corresponding Option (see
Appendix A).
-->
               <OptionName>ECS-TEST PKG1</OptionName>
<!--
Production Option of 'Native Granule' selected
-->
               <ComplexValue>
                  <SimpleOptionSelection>
                     <OptionName>Production Option</OptionName>
                     <Value>Native Granule</Value>
                  </SimpleOptionSelection>
<!--
Media Type choice root
-->
                  <ComplexOptionSelection>
                     <OptionName>Media Type</OptionName>
                     <ComplexValue>
<!--
CDROM media type selected
-->
                        <ComplexOptionSelection>
                           <OptionName>CDROM</OptionName>
                           <ComplexValue>
<!--
Compression type of 'Gzip' selected
-->
                              <SimpleOptionSelection>
                                 <OptionName>Compression</OptionName>
                                 <Value>GZip</Value>
                              </SimpleOptionSelection>
<!--
```

```
DDISTMEDIAFMT value of ‘Rockridge’ selected
-->
                              <SimpleOptionSelection>
                                 <OptionName>DDISTMEDIAFMT</OptionName>
                                 <Value>Rockridge</Value>
                              </SimpleOptionSelection>
                           </ComplexValue>
                        </ComplexOptionSelection>
                     </ComplexValue>
                  </ComplexOptionSelection>
               </ComplexValue>
            </ComplexOptionSelection>
         </OptionSelection>
      </OrderLineItem>
   </CreateOrderRequest>
</OrderEntryService>
```

APPENDIX D: ECHO CLIENT TOOLKIT – THE FACADE

D.1 Client Toolkit

ECHO uses an XML over http RPC style mechanism similar to SOAP and xmlrpc. The primary difference between ECHO and SOAP or xmlrpc is that ECHO has defined a set of DTDs that represent implemented operations. Making use of ECHO requires a client developer to marshal information to and from XML format and then pass the serialized request over a SOAP channel. A significant amount of overhead is incurred by the client developer in order to invoke a remote operation and interpret a response. The client toolkit (aka the façade) has been developed to alleviate some of the burdens client developers currently shoulder.

D.2 XML Example

Assume a developer is implementing a client to the UserAccountService, allowing a user to update their address and telephone information. After the developer collects all relevant address information, they would have to assemble an XML message to send to ECHO. Below is an example of the amount of code required to generate the XML request to add an address to a user's profile:

```
Document requestDoc = new DocumentImpl();
Element service = requestDoc.createElement("UserAccountService");
requestDoc.appendChild(service);

Element transaction = requestDoc.createElement("AddAddressRequest");
service.appendChild(transaction);

Element addressInformationElement = requestDoc.createElement("AddressInformation");
transaction.appendChild(addressInformationElement);

Element idElement = requestDoc.createElement("AddressId");
addressInformationElement.appendChild(idElement);

Element theIdElement = requestDoc.createElement("AddressId");
theIdElement.appendChild(requestDoc.createTextNode(addressId));
IdElement.appendChild(theIdElement);

Element usFormatElement = requestDoc.createElement("USFormat");
usFormatElement.appendChild(requestDoc.createTextNode(usFormat));
addressInformationElement.appendChild(usFormatElement);

//  … and so on for Street1-5, City, State, and Zip Code

// submit the message and get the response
OutputFormat format = new OutputFormat(requestDoc);
String dtdAddress = http://api.echo.eos.nasa.gov/dtd/UserAccountService.dtd;
format.setDoctype(null, dtdAddress);

StringWriter stringWriter = new StringWriter();
XMLSerializer serializer = new XMLSerializer(stringWriter, format);
serializer.asDOMSerializer();
serializer.serialize(requestDoc.getDocumentElement());

String requestMessage = stringWriter.toString();
String responseMessage = null;
try {
  responseMessage = echoToken.perform(requestMessage);
} catch (XMLServiceException e) {
  // do something intelligent
}

// now parse the response message
```

The above code demonstrates how a developer assembles an XML message and submits it to ECHO. It is a relatively simple and straightforward example because it does not contain any hierarchical nested complex structures. Assembling and submitting XML that interacts with the Options framework within ECHO is extremely difficult.

D.3 Client Toolkit Example

In this example, the developer leverages the services of the client toolkit to provide XML message assembly and invocation. Using the toolkit allows the client developer to focus on providing business level functionality and not data marshalling and remote invocation services.

```
AddressInformationDO address = null;
try {
  address = new AddressInformationDO(new AddressID("home"), true, street1, street2, street3,
                                                        street4, street5, city, state, zip, country);
} catch (ValidationException e) {
  // some validation can be performed client side within remote invocation
}

try {
  UserAccountService.AddAddress(address, echoToken);
} catch (XMLServiceException e) {
  // some validation can only be performed on the server, this is where those errors are reported
}
```

Interacting with ECHO via the client toolkit is considered a best practice for java client developers. It reduces your development time, lessens your maintenance obligations, and makes your code significantly more readable. Where possible, use the client toolkit.

D.4 Client Toolkit Limitations

The client toolkit only provides data marshalling for first-class parameters and enumerated types. It does not support data serialization and deserialization of queries and their responses. If you are developing a client that queries ECHO, you can (and should) use the client toolkit for all parameters you pass to the Query transaction except for the actual AQL query. The actual query still needs to be created using a mechanism external to the client toolkit, but it is still recommended that you use the toolkit for assembling all other XML structures external to the query itself.

D.5 Client Toolkit Applicability

The Client Toolkit manages Java to XML translations (and XML to Java translations), and is both stateless and platform independent. A client toolkit is released for each version of ECHO, but there is nothing that prevents a client developer from using version 4.8 of the toolkit to connect to version 5.5 of the ECHO server side software. If the APIs you invoke have changed, the client toolkit will not be able to perform the remote invocation for you. So long as the APIs are the same, the toolkit can talk to any ECHO version or platform.